Hornig/Trapp/Weltner

Tips en Trucs voor de Commodore 64 deel 2

De trucendoos opnieuw geopend

Commodore Bibliotheek 12

DATA BECKER NEDERLANDS*

Tips en Trucs voor de Commodore 64 deel 2

Hornig/Trapp/Weltner

Tips en Trucs voor de Commodore 64 deel 2

De trucendoos opnieuw geopend

DATA BECKER
NEDERLANDS*

Oorspronkelijke titel: Noch mehr Tips & Tricks Band 2
Copyright © 1984 Data Becker GmbH, Düsseldorf
Voor de Nederlandse vertaling:
Copyright © 1985 Uitg. A.W. Bruna & Zoon, Utrecht
Basisvormgeving omslag: Studio AWB/Martin van Keulen
Vertaling: Jaap Bloem
Produktie: ProCread, Groningen
Redaktie: ProCread, Groningen
ISBN 90 229 3337 7
D/1986/0939/194

De keuze en opbouw van de programma's die in dit boek voorkomen, zijn gebaseerd op hun educatieve waarde. Alle programma's zijn getest op hun juiste werking. De uitgever kan geen enkele verantwoordelijkheid voor eventueel toch optredende fouten aanvaarden.

Uit deze uitgave mag niets worden verveelvoudigd en/of openbaar gemaakt door middel van druk, fotokopie, microfilm of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

Boeken en programma's van

DATA BECKER
NEDERLANDS*

worden in de handel gebracht door:
A.W. Bruna & Zoons Uitgeversmij b.v.
Postbus 8411, 3503 RK Utrecht
en
A.W. Bruna & Zoon n.v.
Antwerpsesteenweg 29a, 2630 Aartselaar.

1 **INLEIDING** 9

2 **TIPS & TRUCS VOOR THUIS**
2.1 Het sturen van de datasette in BASIC 10
2.2 Een kopieerbeveiliging voor de datasette 11
2.3 Het verschuiven van de cassettebuffer 11
2.4 Alleen laden met code 12
2.5 Foutkanaal uitlezen 13
2.6 SAVE wordt LOAD 13
2.7 Automatisch naladen 14
2.8 LOAD en SAVE bij programma's in machinetaal 15
2.9 ASCII-code veranderen in beeldschermcode 16
2.10 Invoer van hexadecimale getallen 18
2.11 De DATA-generator 19
2.12 Screencopy 20
2.13 BASIC-tips 22
2.13.1 Gebruik geen nul, maar een punt 22
2.13.2 Gebruik geen ASCII-tekens, maar variabelen 23
2.13.3 Gebruik geen integere, maar reële variabelen 24
2.13.4 Gebruik FOR-NEXT-lussen 25
2.13.5 Zet subroutines aan het begin van het programma 25
2.14 De ESCAPE-functie 26
2.15 Beeldschermkleuren veranderen 28
2.16 Twee beeldschermen 29
2.17 Lichtkrant in machinetaal 32
2.18 De STOP-functie 34
2.19 Toeval? Opmerkingen over RND 36
2.20 INPUT-variaties 39
2.21 Trucs met de diskdrive 41

3 **BEVEILIGING VAN SOFTWARE**
3.1 De LIST-functie 46
3.2 Listen zonder regelnummers 46
3.3 LIST-beveiliging - LIST-functie uitschakelen 47
3.4 Veranderen van de BASIC-LINK 49
3.5 Regels wissen? SYNTAX ERROR! 51
3.6 Kunstmatige stuurtekens 52
3.7 Beveiliging met POKEs 53
3.8 Het blokkeren van gevaarlijke toetsen 54
3.9 Machinetaal ... hoewel? 55

4 **ZELF INSTRUCTIES UITBREIDEN**
4.1 BASIC-LINKS veranderen 57
4.2 CHRGET-routine veranderen 59
4.3 IRQ-vector veranderen 61

5 GRAFIEK

5.1 Basiskennis 63
5.2 De karaktergenerator in het geheugen 65
5.3 Karakterset uitlezen 65
5.4 Karaktergenerator kopiëren 67
5.5 Karaktergenerator omschakelen 68
5.6 Hulpprogrammatuur voor de definitie van karaktersets 70
5.7 DESIGN in de listing 83
5.8 MULTI COLOR 85
5.9 Transparanten printen zonder moeite 87
5.10 Een grote lichtkrant 88
5.11 Acht blocks voor sprites 89

6 HET SPEL 91

6.1 Het gèraamte 92
6.2 Grafiek 97
6.3 Geluid 99
6.4 De gebruiksaanwijzing 100
6.5 Beginbeeld 101
6.6 Samenvatting 102

7 INTERRUPTS

7.1 RESET 108
7.2 NMI 110
7.3 IRQ 112
7.4 Het programmeren van de IRQ 115
7.5 Toetsenbordpiep 117
7.6 Muziek 118

8 BEDRIJFSSYSTEEM: ROM IN RAM

8.1 Kopieerroutines 120

9 ROUTINES VAN HET BEDRIJFSSYSTEEM 123

10 KERNAL 134

11 HET GEHEUGEN

11.1 Het opslaan van BASIC-regels 151
11.2 De BASIC-monitor 153
11.3 De zero page 156
11.4 Belangrijke adressen van de volgende pages 163
11.5 Het opslaan van variabelen 164
11.6 Lijst van belangrijke wijzers 166

12 BIJLAGE

12.1 De tabellen 168
12.2 Omrekeningstabel 169
12.3 Tabel van divicenummers 178

1 INLEIDING

In de Commodore 64 zitten enkele schoonheidsfoutjes:

(1) Het gelijktijdig indrukken van de linker SHIFT-toets en de cursortoetsen resulteert op het beeldscherm in het schoppenteken (SHIFT en A).

(2) Door de Commodore-toets, puntkomma en = in te drukken verdwijnt de cursor. POKE 53281,0 vertelt u waarom: de witte kleur van de karakters is veranderd in blauw.

(3) Ga met de cursor naar de onderste regel van het scherm en schrijf tot de cursor voor de tweede keer uit het zicht is verdwenen (dit geldt ook voor spaties). Probeert u met de INST/DEL-toets het 80e teken te wissen, dan treedt het euvel op. Op het scherm verschijnt het volgende beeld:

```
LOAD

?SYNTAX ERROR
READY.
RUN

READY.
```

Nu gaat alles mis. Een BASIC-programma hangt zich op bij de eerste INPUT-instructie, omdat het toetsenbord niet functioneert. Om uit deze situatie te komen kunt u behalve resetten of de computer uitschakelen ook #(SHIFT+3) inlezen. De computer reageert met PRESS PLAY ON TAPE. Volg dat advies op, als er een cassetterecorder is aangesloten. De Commodore 64 gaat in de laad-modus, die met RUN/STOP kan worden onderbroken. Deze fout, die alleen bij oudere modellen optreedt, komt voor als de kleur van de karakters rood, cyaan, blauw, geel (variant van de fout), lichtrood, grijs 1, lichtblauw of grijs 3 is. Het gaat dus om de toetsen 3, 4, 7 en 8 in combinatie met CTRL of C=. Bij de nieuwe versies van de Commodore 64 verschijnt een A in de linker bovenhoek, als na het wissen van het scherm POKE 1024,1 wordt ingelezen. Bij oudere machines moet bovendien nog een waarde in het kleur-RAM worden gePOKEt om een teken te laten verschijnen.

De oorzaak van de mankementen is ons onbekend. De eerste twee hangen waarschijnlijk samen met het uitlezen van het toetsenbord. Aan het laatste ligt waarschijnlijk een ongegrond verklaarde IRQ-routine ten grondslag: TI$ wordt verder gezet, de cursor knippert, enzovoort.

2 TIPS & TRUCS VOOR THUIS

De volgende bladzijden gaan over nuttige routines in BASIC en machinetaal en de mogelijkheden van interessante geheugenadressen.

2.1 Het sturen van de datasette in BASIC

Na beëindiging van een LOAD- of SAVE-actie zorgt de computer ervoor, dat de motor stopt. Pas nadat STOP is ingedrukt kan de motor weer met PLAY worden gestart. Adres 1 (processor-poort) en de flag voor de cassettemotor (adres 192, $C0) zijn van belang voor de softwarematige besturing van de cassettemotor. Een voorbeeld: druk op PLAY en typ

```
POKE 192,1: POKE 1,PEEK(1) OR 32
```

in in de direkte modus. De vijfde bit van adres 1 en de flag voor de cassettemotor worden geset: de motor stopt. Met

```
POKE 1,PEEK(1) AND 39: POKE 192,0
```

start u de motor weer. Via adres 1 kan ook informatie worden gegeven over de stand van de toetsen:

```
IF PEEK(1)=55 THEN PRINT "GEEN TOETS INGEDRUKT"
IF PEEK(1)=7  THEN PRINT "TOETS INGEDRUKT"
```

Dit kunt u met de WAIT-instructie gebruiken:

```
WAIT 1,16
```

heeft het resultaat dat er wordt gewacht tot de stoptoets van de recorder is ingedrukt.

Samengevat:

recordermotor uit	POKE 192,1: POKE 1,PEEK(1) or 32
recordermotor aan	POKE 1, PEEK(1) AND 39: POKE 192,0
wachten op STOP-toets	WAIT 1,16
wachten op PLAY-toets	xxx IF PEEK(1)=55 THEN GOTO xxx

2.2 Een kopieerbeveiliging voor de datasette

De naam van een programma op cassette mag 172 tekens lang zijn. Wordt het programma in de computer geladen, dan komen er ten hoogste 16 karakters op het scherm te staan. De overige tekens blijven in de cassettebuffer. Het BASIC-programma hieronder leest het onbekende zeventiende teken van de programmanaam in.

```
10 IF PEEK(849)=ASC("X") THEN GOTO 30
20 PRINT "? LOAD ERROR": NEW
30 PRINT "OORSPRONKELIJKE PROGRAMMA"
```

Zet deze regels voor het eigen programma en kies, als u het opslaat, een naam met een willekeurig zeventiende teken:

```
SAVE "1234567890123456X"
```

Als het programma wordt gekopieerd, ontbreekt het laatste teken. De computer geeft een ?LOAD ERROR en de programmawijzers nemen hun startpositie in. Het programma kan alleen met een RENEW alsnog worden gerund (zie PEEKS EN POKES uit de serie DATA BECKER NEDERLANDS*). Op deze manier is niet te zien, dat er een kopieerbeveiliging is gebruikt. Een nadeel is, dat elk programma op cassette zonder meer wordt geladen als het alleen met LOAD wordt aangeroepen. Door de drie BASIC-regels van een LIST-beveiliging te voorzien, wordt de kopieerbeveiliging aanmerkelijk beter (zie hoofdstuk 3).

2.3 Het verschuiven van de cassettebuffer

In de cassettebuffer (adresbereik $033C-$03FB, 828-1019) worden de gegevens van of voor de band tussentijds opgeslagen en in blokken uitgevoerd. Omdat de cassettebuffer alleen bij LOAD- en SAVE-instructies wordt gebruikt, kunnen daar kleine programma's in machinetaal worden opgeslagen of de sprite-blocks 13-15. U kunt voorkomen, dat bij een LOAD- of SAVE-actie over de gegevens in de buffer heen wordt geschreven door de vector van de cassettebuffer ($B2-$B3, 178-179) bijvoorbeeld in het ongebruikte geheugenbereik vanaf $C000 te zetten. Gewoonlijk wijst de vector op het begin van de buffer.

```
10 A=49152: REM beginadres nieuwe cassettebuffer
20 HI=INT(A-256)
30 LO=A-(HI*256)
40 POKE 178,LO: POKE 179,HI
```

Is het C-bereik in gebruik, dan kunt u de wijk nemen naar het beeldschermgeheugen, dat bij een datasette-LOAD- of -SAVE-actie toch niet wordt gebruikt. Vervang regel 10 dan door:

```
10 A=1024
```

2.4 Alleen laden met code

Als u de diskdrive 1541 gebruikt, wordt de naam van het programma bij LOAD- en SAVE-instructies ook in het geheugen opgeslagen. Wijzer $BB-$BC (187-188) geeft aan of dat is gebeurd. Het volgende BASIC-programma laat bij diskette-operaties de naam van het laatste programma op het scherm verschijnen:

```
10 AD=PEEK(187)+256*PEEK(188)
20 FOR A=AD TO 40959: B=PEEK(A)
30 NA$=CHR$(B)
40 NEXT: PRINT NA$
```

Laat de volgende regels aan een programma voorafgaan:

```
10 FOR A=40955 TO 40959
11 B=PEEK(A)
12 READ C$: C=ASC(C$)
13 IFB=C THEN 15
14 PRINT "?LOAD ERROR";: NEW
15 NEXT
16 DATA C,O,D,E,1
```

Sla het programma vervolgens op onder een naam van 16 tekens. Als het wordt geladen en gestart, verschijnt een gesimuleerde LOAD ERROR net als in paragraaf 2.2. Het programma is dus niet gewist, maar de programmawijzers nemen hun uitgangspositie in. U voorkomt problemen door bij het laden de naam van het programma met CODE1 uit te breiden. Regel 10 leest de laatste vijf tekens van de naam in. Pas als die gelijk zijn, loopt het programma zonder oponthoud. De naam moet 16 tekens lang zijn, anders kunt u geen andere code toevoegen.

2.5 Foutkanaal uitlezen

Een fout bij het gebruik van diskdrive 1541 resulteert in het knipperen van de rode LED. De volgende routine leest het foutkanaal van de drive uit en geeft de foutmelding:

```
10 REM foutkanaal van floppy uitlezen
20 OPEN 1,8,15: REM kanaal openen
30 INPUT #1,F,FM$,SP,SE
40 PRINT "F","FM$","SP","SE"
50 CLOSE 1
```

variabelen:

F - foutnummer
FM$- foutmelding
SP - spoor
SE - sector

2.6 SAVE wordt LOAD

De vectoren in deze paragraaf en hun manipulatie zijn interessant voor datasette en diskdrive. De SAVE- ($0332-$0333, 818-819) en de LOAD-vector ($0330-$0331, 816-817) wijzen op routines in het ROM. SAVE wordt LOAD (VERIFY) door

```
POKE 818, PEEK(816): POKE 819, PEEK(817)
```

Er kan niet meer zomaar worden gesaved.

```
POKE 818,237: POKE 819,245  (wijzer $F5ED)
```

herstelt de oude toestand. Door

```
POKE 816, PEEK(818): POKE 817, PEEK(819)
```

wordt LOAD uitgeschakeld. Nu brengt

```
POKE 816,165: POKE 817,144 (wijzer $F4A5)
```

u weer in de oude toestand terug. LOAD en SAVE verwisselen gaat ook heel gemakkelijk:

```
POKE 816,237: POKE 17,245: POKE 818,165: POKE 819,244

POKE 818(816),26: POKE 19(817),167
```

schakelt LOAD en SAVE uit. Beide instructies worden beantwoord met READY.

2.7 Automatisch naladen

Een programma, dat ondertussen andere delen nalaadt of bestanden gebruikt, levert geen problemen op, als bekend is of de diskdrive of de datasette wordt gebruikt.

```
450 FD=PEEK(186): OPEN 1,FD,1
```

of

```
670 FD=PEEK(186): LOAD "TEST",FD,(secundaire adres)
```

geeft aan welk apparaat in bedrijf is. Er wordt van uitgegaan, dat de manier, waarop het hoofdprogramma werd geladen, ook voor de verdere opslag geldt. Op adres 186 staat het nummer (primaire adres) van het apparaat, dat het laatst werd gebruikt. U hoeft niet op de printer te letten, als aan het begin van het programma variabele FD de inhoud van adres 186 toewijst.

Als het laden van een cassetteprogramma LOAD ERROR tot gevolg heeft, betekent dat niet dat het verloren is. De VIC 20 en de Commodore 64 zetten een programma namelijk twee keer op de band. Bij het laden wordt de eerste versie in de buffer vergeleken met de tweede op de band. Een verschil leidt tot LOAD ERROR. Kan het programma worden gelist, dan is er niets aan de hand. Pas als het programma foutloos is geladen, neemt de computer de lengte over (adres 831-832 in de cassettebuffer). Bij LOAD ERROR ontbreekt die informatie. De variabelestart ligt in het BASIC-programma en vernietigt het.

In tegenstelling tot paragraaf 2.2 en 2.4 gaat het hier om een echte LOAD ERROR; van simulatie is geen sprake. In de direkte modus ingelezen, redden de volgende regels een verkeerd geladen programma als het kan worden gelist:

```
POKE 46,PEEK(832):
POKE 47,PEEK(831):
POKE 48,PEEK(832):
POKE 49,PEEK(831):
POKE 50,PEEK(832)
```

U start het programma met RUN.

2.8 LOAD en SAVE bij programma's in machinetaal

Een machinetaalprogramma direkt op band zetten gaat sneller dan met de BASIC-loader en bespaart de geheugenruimte van de BASIC-loader. Op die manier kunt u bijvoorbeeld makkelijk zelfgedefinieerde tekens op cassette opslaan. Bij een SAVE-actie wordt alles tussen de BASIC-start (43-44) en het begin van de variabelen (45-46) op band gezet. Als $5000-$6000 wordt gesaved, schrijf dan het beginadres in vector 43-44 en het eindadres in 45-46.

```
$5000=#20480  LB=0  HB=80
$6000=#24576  LB=0  HB=96
```

Sla het bereik op met

```
POKE 43,0: POKE 44,80: POKE 45,0: POKE 46,96: CLR:
SAVE "(naam)",1,1
```

De eerste 1 achter SAVE is het primaire adres van het apparaat (8 is diskdrive). De tweede 1 betekent, dat er direkt wordt geladen: als u opnieuw laadt, gebeurt dat automatisch op hetzelfde adres. Bovendien verslijt de computer het bereik niet voor een BASIC-programma en berekent daarom geen links. Een programma, dat is opgeslagen in $5000-$6000, laadt u in $8000-$9000 met

```
SYS 63276 ($F72C)
```

Op het scherm verschijnt PRESS PLAY ON TAPE. Als u dat doet, wordt alleen de tape-header geladen. Omdat daarin de plaats van het programma staat, kan het gemakkelijk worden verschoven.

```
POKE 829,LB (begin)
POKE 830,HB (begin)
POKE 831,LB (einde+2)
POKE 832,HB (einde)
```

Lees voor $8000-$9000 in:

```
POKE 829,0: POKE 830,128: POKE 831,0: POKE 832,144
SYS 62849 ($F581)
```

De laatste instructie hervat de LOAD-actie. Geef na beëindiging NEW om te starten. Omdat de diskdrive de cassettebuffer niet gebruikt, kunnen alleen met de datasette op deze manier adresssen worden veranderd.

2.9 ASCII-code veranderen in beeldschermcode

In de bijlage van het handboek voor de Commodore 64 staan tabellen voor ASCII- en beeldschermcode. Met de beeldschermcode worden tekens in het beeldschermgeheugen gePOKEt ($0400, 1024). De ASCII-code kan worden veranderd in een string, die met PRINT kan worden uitgevoerd (ASC-CHR$). De tekens in beide tabellen zijn hetzelfde, alleen de codes verschillen. In sommige programma's moeten de ASCII-codes worden veranderd in beeldschermcodes (zie bijvoorbeeld de listing van de teken-editor). De volgende BASIC-routine zorgt daarvoor:

```
10 REM ASCII- in beeldschermcode veranderen
20 T$="A": T=ASC(T$): PRINT "ASCII-CODE="; T
30 PRINT "(home)"; T$
40 BS=256*PEEK(648): REM BS = begin beeldschermcode
50 CO=PEEK(BS): REM CO = beeldschermcode
60 PRINT "BEELDSCHERMCODE="; CO
70 END
```

Stringvariabele T$ bevat het teken, hier de letter A. ASC vraagt de ASCII-waarde op en voert die uit. Om achter de beeldschermcode te komen, wordt eerst het beginadres van het beeldschermgeheugen berekend (adres 648 = hibyte van video-RAM). De letter in T$ wordt in de eerste geheugenplaats van het beeldschermgeheugen geprint (regel 30), die met PEEK wordt ingelezen. Zo krijgt u de beeldschermcode. Het gebruik van het beeldschermgeheugen heeft als resultaat, dat het teken linksboven op het scherm verschijnt.

De volgende hulproutines gaan het probleem op rekenkundige wijz te lijf.

```
        10 REM video- in ASCII-code veranderen
        20 INPUT "WAT IS DE BEELDSCHERMCODE"; CO
        30 CO=CO AND 127
40 IF CO AND 32 THEN 7        0
50 IF CO AND 64 THEN AC=CO OR 32: GOTO 90
60 AC=CO OR 64: GOTO 90
70 IF CO AND 64 THEN AC=CO AND 63 OR 12: GOTO 90
80 AC=CO
90 PRINT "ASCII-CODE="; AC
100 END
```

```
10 REM ASCII- in video-code veranderen
20 INPUT "WAT IS DE ASCII-CODE"; AC
30 IF AC AND 128 THEN CO=AC AND 127 OR 64: GOTO 70
40 IF NOT AC AND 64 THEN CO=AC: GOTO 70
50 IF AC AND 32 THEN CO=AC AND 95: GOTO 70
60 CO=AC AND 63
70 PRINT "VIDEO-CODE="; CO
```

variabelen:

AC = ASCII-waarde
CO = videocode

Met een PRINT-instructie en de cursorbesturing kunnen natuurlijk de delen van het scherm worden gewist, die een programma verlangt. Overzichtelijker en eenvoudiger gaat het met een routine uit de ROM: $E9FF (59903) wissen van een beeldschermregel, die de positie van de cursor niet aantast. Cursorsturing is overbodig. In BASIC roept u de routine op met

```
POKE 781,x: SYS 59903
```

x = 0-24

De regel (0-24) wordt aan het X-register toegewezen. De routine doet de rest. Door de eerste twee bytes over te slaan (het nieuwe sprongadres wordt $EA01, 59905) kan het aantal te wissen tekens in regel x in het Z-register (adres 782) worden gezet.

```
POKE 781,x: POKE 782,z: SYS 59905
```

x = 0-24, z = 0-39 (0-255)

2.10 Invoer van hexadecimale getallen

Machinetaalprogramma's opslaan in het geheugen kan gebeuren door

a) monitor
b) BASIC-loader
c) hexadecimale getallen

Het volgende programma laadt hexadecimale getallen. De computer vraagt eerst naar de stapgrootte, het aantal hexadecimale getallen per regel. Dat mag in dit programma niet groter zijn dan 11, maar dat kunt u natuurlijk veranderen. Het beginadres moet in hexadecimale vorm worden ingevoerd en uit 4 tekens bestaan. De computer geeft vervolgens de geheugenplaats van het eerste element van elke regel; u kunt de getallen invoeren. Met RETURN breekt u het programma af.

```
0 PRINT"■■            HEX-INVOER-PROGRAMMA"
10 INPUT"■■■■STAPGROOTTE:";SW:IFSW<1ORSW>
11THEN10
20 INPUT"BEGINADRES:";A$
30 IFLEN(A$)<>4THENPRINT"VERKEERDE INVOE
R":GOTO20
40 POKE19,0:PRINT:Y=3:FORZ=1TO4:B$=MID$(
A$,Z,1):B=ASC(B$)
50 IFB>47ANDB<58THENSA=SA+VAL(B$)*16↑Y:G
OTO80
60 IFB>64ANDB<71THENSA=SA+(ASC(B$)-55)*1
6↑Y:GOTO80
70 PRINT"VERKEERDE INVOER":GOTO20
80 Y=Y-1:NEXTZ
100 PRINT"■"SA;:FORZ1=0TOSW-1:FORZ2=1TO0
STEP-1
110 POKE204,0:GETA$:IFA$=""THEN110
120 POKE207,0:B=ASC(A$):IFB>47ANDB<58THE
NPE=PE+VAL(A$)*16↑Z2:GOTO160
130 IFB>64ANDB<71THENPE=PE+(ASC(A$)-55)*
16↑Z2:GOTO160
140 IFB=13THENEND
150 GOTO110
160 PRINTA$;:NEXTZ2:PRINT" ";:POKESA+Z1,
PE:PE=0:NEXTZ1
170 SA=SA+SW:PRINT:GOTO100
```

2.11 De DATA-generator

Dit programma vertaalt een machinetaalprogramma in het geheugen in DATA-regels. Geef het begin en de lengte van het machinetaalprogramma aan en het eerste regelnummer van de DATA-regels. Dit nummer mag niet kleiner zijn dan 221, anders schrijft het programma over zichzelf heen. Omdat het programma zichzelf na elke DATA-regel onderbreekt, moesten de variabelen in speciale geheugenplaatsen worden ondergebracht.

828-829 - begin van machinetaalprogramma
830-831 - lengte van machinetaalprogramma
832-833 - actuele regelnummer

Schrijf CLR-HOME en RETURN in de buffer voor het toetsenbord en de DATA-regel wordt in het programma overgenomen. Na RUN 100 en RETURN begint het programma bij regel 100. In regel 15 staan de waarden voor de toetsenbordbuffer. Regel 20 berekent LSB en MSB. U kunt het programma zo veranderen, dat variabelen in DATA-regels worden ondergebracht en bij elke doorloop worden gelezen.

```
10 GOTO25
15 DATA19,13,82,213,49,48,48,13
20 HB=INT(X1/256):LB=X1-256*HB:RETURN
25 PRINT"▌██████████▌ DATA - GENERATOR  █
"
26 PRINT"██GEEF DE VOLGENDE PARAMETERS I
N:███"
30 INPUT"█PROGRAMMA-BEGIN";PB
40 INPUT"█PROGRAMMA-LENGTE";PL
50 INPUT"█BEGIN-REGEL";BR
60 X1=PB-1:GOSUB20:POKE828,LB:POKE829,HB
70 X1=PL:GOSUB20:POKE830,LB:POKE831,HB
80 X1=BR:GOSUB20:POKE832,LB:POKE833,HB
100 X=PEEK(832)+256*PEEK(833)
110 Y=PEEK(828)+256*PEEK(829)
120 Z=PEEK(830)+256*PEEK(831)
130 PRINT"▌"X" DATA";
140 FORZ1=1TO10:PRINTPEEK(Y+Z1)"█,";
150 Z=Z-1:IFZ=0THENPRINT"█ ":GOTO220
160 NEXTZ1:PRINT"█ "
170 X=X+1:Y=Y+10
180 X1=Y:GOSUB20:POKE828,LB:POKE829,HB
190 X1=Z:GOSUB20:POKE830,LB:POKE831,HB
```

```
200 X1=X:GOSUB20:POKE832,LB:POKE833,HB
210 FORZ1=631TO638:READA:POKEZ1,A:NEXTZ1
:POKE198,8:END
220 POKE631,19:POKE632,13:POKE198,2:END
```

2.12 Screencopy

Met het screencopy-programma behoort de onoverzichtelijkheid en de relatieve traagheid van PRINT-instructies tot het verleden. Nadat u het met RUN (!) hebt gestart, verschijnt de cursor en kan het beeldschermmasker worden gemaakt. Ga als u klaar bent naar de laatste regel en druk op RETURN. De computer vraagt vervolgens naar de beginregel. U geeft het getal. Daarna voert hij de inhoud van het beeldscherm uit en neemt het regel voor regel over in het programma. Wat op het scherm stond heeft na beëindiging van het programma de vorm van PRINT-instructies. Daarna kunt u een volgend beeld maken, dat u onder een ander regelnummer opslaat. U kunt ook sreencopy wissen en alleen de PRINT-instructies laten staan. Met het volgende programma gaat dat handiger, dan dat na elk ingetoetst regelnummer RETURN moet worden geven.

```
0 GOTO 27
26 POKE 254,1
27 X=PEEK(254):PRINT"]"X
28 X=X+1:POKE254,X:IFX=27 THEN PRINT"SCR
EENCOPY GEWIST":END
29 POKE 631,19:POKE632,13:POKE633,82:POK
E634,213:POKE635,13:POKE198,5:END
```

U moet met de hand alleen nog regel 0, 26, 27, 28 en 29 wissen. PRINT-regels zijn heel geschikt om beelden te maken, maar hebben het nadeel, dat verandering van de kleur van de cursor, RVS ON en RVS OFF achteraf moeten worden toegevoegd. Daar is wel wat aan te doen, maar het kost meer moeite dan het oplevert.

```
1 GOTO10
2 PRINT"♥":OPEN1,0
3 INPUT#1,A$:IFPEEK(214)=24THEN5
4 PRINT:GOTO3
5 FORZ=0TO999:POKE24576+Z,PEEK(1024+Z):N
EXTZ
6 INPUT"♥BEGIN-REGEL";BR:IFBR<30THEN6
7 HB=INT(BR/256):LB=BR-256*HB
8 POKE828,LB:POKE829,HB
9 POKE830,0:POKE831,96
10 SP=PEEK(830)+256*PEEK(831)
11 FORZ=0TO39:A=PEEK(SP+Z)
12 GOSUB21
13 AU$=AU$+CHR$(B):NEXTZ
14 ZE=PEEK(828)+256*PEEK(829)
15 PRINT"♥"ZE"?"CHR$(34);AU$;CHR$(34)";"
16 SP=SP+40:IFSP>25536THEN20
17 HB=INT(SP/256):LB=SP-256*HB:POKE830,L
B:POKE831,HB
18 ZE=ZE+10:HB=INT(ZE/256):LB=ZE-256*HB:
POKE828,LB:POKE829,HB
19 POKE631,19:POKE632,13:POKE633,82:POKE
634,213:POKE635,13:POKE198,5:END
20 POKE631,19:POKE632,13:POKE198,2:END
21 A=AAND127:IFAAND32THEN24
22 IFAAND64THENB=AOR32:RETURN
23 B=AOR64:RETURN
24 IFAAND64THENB=AAND63OR128:RETURN
25 B=A:RETURN
```

Omdat dit geen eenvoudig programma is, geven we hier enkele verklaringen.

regel 2-4	Het beeldscherm wordt als bestand geopend. Bij RETURN wordt in regel 3 bekeken of de cursor op de onderste regel staat.
regel 5	De inhoud van het scherm wordt opgeslagen, omdat het bij het verdere verloop verandert.
regel 8	Het nummer van de beginregel wordt ingevoerd, gecontroleerd, gesplitst in LSB en MSB en weggezet in de adressen 828-829 (bandbuffer).
regel 9	LSB en MSB van het adres, waarop de opgeslagen inhoud van het scherm begint.
regel 13	De string voor de regel wordt samengesteld.

regel 14 — Het regelnummer wordt berekend.

regel 15 — De uiteindelijke regel verschijnt op het scherm.

regel 16-18 — Het nieuwe regelnummer verschijnt op het scherm.

regel 16-18 — Het nieuwe regelnummer en het nieuwe adres van de beeldscherminhoud worden berekend, gesplitst in LSB en MSB en zo opgeslagen, dat het programma zichzelf onderbreekt en dat bij een nieuwe start de variabelen zouden zijn gewist.

regel 19 — Het programma onderbreekt zichzelf en neemt de regel (zie regel 15) op doordat de cursor naar de HOME-positie springt en een RETURN wordt gegeven. Het programma wordt met RUN automatisch gestart. De operaties vinden plaats door de ASCII-waarden in de toetsenbordbuffer te poken.

regel 21-25 — Een subroutine, die beeldscherm- in ASCII-codes verandert. De waarden op het scherm en in het geheugen zijn videocodes, maar de gegenereerde beeldschermregel (AU$) verlangt ASCII-waarden. U kunt deze subroutine ook voor andere doeleinden gebruiken.

2.13 BASIC-tips

Adviezen om de snelheid van het tamelijk trage BASIC te optimaliseren vinden waarschijnlijk bij elke programmeur een welwillend gehoor.

2.13.1 Gebruik geen nul, maar een punt

Een (alleenstaande) decimale punt in een BASIC-programma wordt automatisch als het reëele getal 0 geïnterpreteerd. De nul in een BASIC-programma is voor de computer geen getal, maar een ASCII-teken, dat eerst in een getal moet worden veranderd. Bij een decimale punt hoeft dat niet. Dat spaart dus tijd.

Voorbeeld:

programma 1:

```
10 TI$="000000"
20 FOR X=1 TO 100
30 A=A+0
40 40 NEXT Z
50 PRINT TI
```

programma 2:

```
10 TI$="000000"
20 FOR=1 TO 100
30 A=A+.
40 NEXT X
50 PRINT TI
```

Nog vaker kan de decimale punt in IF-instructies worden gebruikt.

In plaats van

```
10 IF A=0 THEN 20
```

gebruikt u

```
10 IF A=. THEN 20
```

2.13.2 Gebruik geen ASCII-tekens, maar variabelen

Een als volgt geschreven getal wordt in het BASIC-geheugen als ASCII-waarde opgeslagen:

```
10 POKE 1024,1
```

Bij het uitvoeren van de instructie moet de computer deze ASCII-string eerst in een integer getal en daarna in een reëel getal veranderen. Hij rekent uitsluitend met reële getallen. Door een variabele te gebruiken kan de ASCII-string als reëel getal worden opgeslagen.

Voorbeeld:

programma 1:

```
10 TI$="000000"
20 FOR X=1 TO 100
30 A=A+100
```

```
40 NEXT X
50 PRINT TI
```

programma 2:

```
10 TI$="000000"
20 FOR X=1 TO 100
30 A=A+B
40 NEXT X
50 PRINT TI
```

Zeker bij grotere getallen is het verschil aanzienlijk, omdat de ASCII-string, die moet worden veranderd steeds langer wordt. De lengte van het reële getal blijft gelijk; alleen de waarden van de bytes worden anders.

2.13.3 Gebruik geen integere, maar reële variabelen

Omdat, zoals we zagen, de verandering naar reëel veel tijd kost, kunt u het beste ook meteen reële variabelen gebruiken.

Voorbeeld:

programma 1:

```
10 TI$="000000"
20 B%=100:C%=50:D%=25
30 FOR X=1 TO 100
40 A%=A%+B%-C%-D%
50 NEXT X
60 PRINT TI
```

programma 2:

```
10 TI$="000000"
20 B=100:C=50:D=25
30 FOR X=1 TO 100
40 A=A+B-C-D
50 NEXT X
60 PRINT TI
```

2.13.4 Gebruik FOR-NEXT-lussen

FOR-NEXT-lussen zijn sneller en eenvoudiger te programmeren dan IF-instructies.

Voorbeeld:

programma 1:

```
10 TI$="000000":X=1
20 A=A+X
30 X=X+1
40 PRINT TI
```

programma 2:

```
10 TI$="000000"
20 FOR X=1 TO 100
30 A=A+X
40 NEXT X
50 PRINT II
```

2.13.5 Zet subroutines aan het begin van het programma

De computer zoekt vanaf de BASIC-start naar een bepaald regelnummer. Zet subroutines dus daar neer.

Voorbeeld:

programma 1:

```
 10 TI$="000000"
 20 FOR X=1 TO 100
 30 GOSUB 100
 40 NEXT X
 50 PRINT TI
 60 END
100 A=A+X
110 RETURN
```

programma 2:

```
 10 A=A+Z
 20 RETURN
100 TI$="000000"
110 FOR X=1 TO 100
120 GOSUB 10
```

```
130 NEXT X
140 PRINT TI
```

Met RUN 100 start u het tweede programma. Het verschil is hier nog niet zo groot, omdat de programma's vrij kort zijn.

2.14 De ESCAPE-functie

Als u in de aanhalingsmodus (quote mode: "...") bijvoorbeeld de cursorbesturing gebruikt, verschijnen de stuurtekens van de cursor op het scherm. Dat geldt ook voor de kleurtoetsen. Pas bij een PRINT-instructie is de functie van de stuurtoetsen relevant. Het vergt veel tijd om uit deze modus te komen, wanneer bijvoorbeeld een foute invoer moet worden gecorrigeerd. F1 schakelt in een keer terug naar de normale modus, met F2 zit u weer in de aanhalingsmodus.Het aanhalingsteken in de rechter bovenhoek van het beeldscherm geeft de aanhalingsmodus aan.

```
  0 REM ESCAPE
 10 FOR I=40704 TO 40795
 20 READ A
 30 S=S+A
 40 POKE I,A
 50 NEXT
 60 IF S<>1 1000 THEN PRINT "FOUT IN DATA" : END
 70 PRINT "DATA OK"
 80 SYS 40704
 90 DATA 120,169, 15,141, 20,  3,169,159,141
100 DATA  21,  3,133, 56, 88, 96, 72,165,203
110 DATA 212,  4,240, 34,201,  5,240, 59,165
120 DATA 212,208,  4,165,216,240, 13,169, 34
130 DATA 141, 39,  4,169,  1,141, 39,216, 76
140 DATA  52,159,169, 32,141, 39,  4,104, 76
150 DATA  49,234,165,212,240, 18,169,  0,133
160 DATA 212,133,216,169, 20,141,119,  2,169
170 DATA   1,133,198, 76, 52,159,165,216,240
180 DATA 22806,76,60,159,169,255,133,216, 76
190 DATA  52,159
```

De machinetaal-listing ziet er als volgt uit:

```
9F00  SEI
9F01  LDA #$0F
```

```
9F03  STA $0314
9F06  LDA #$9F
9F08  STA $0315
9F0D  CLI
9F0D  RTS
9F0E  PHA
9F0F  LDA
9F10  LDA $CB
9F12  CMP #$04
9F14  BEQ 9F38
9F16  CMP #$05
9F18  BEQ $9F55
9F1A  LDA $D4
9F1C  BNE $9F22
9F1E  LDA $D8
9F20  BEQ $9F2F
9F22  LDA #$22
9F24  STA $0427
9F27  LDA #$01
9F29  STA #D827
9F2C  JMP $9F34
9F2F  LDA #$20
9F31  STA $0427
9F34  PLA
9F35  JMP $EA31
9F38  LDA $D4
9F3A  BEQ $9F4E
9F3C  LDA #$00
9F3E  STA $D4
9F40  STA $D8
9F42  LDA #$14
9F44  STA $0277
9F47  LDA #$01
9F49  STA $C6
9F4B  JMP $9F34
9F4E  LDA $D8
9F50  BEQ $9F34
9F52  JMP $9F37
9F55  LDA #$FF
9F57  STA $D8
9F59  JMP $9F34
```

2.15 Beeldschermkleuren veranderen

Vooral bij zwart-wit apparaten is er vaak op het scherm iets niet te lezen, omdat de kleurencombinatie van achtergrond, kader en schrift verkeerd is gekozen. Het programma in deze paragraaf kan in enkele machinetaalprogramma's en (bijna) alle BASIC-programma's de kleur van achtergrond en kader veranderen in de direkte modus. F1 verandert de kleur van het kader, F3 die van de achtergrond. Daarbij gaat het beeldscherm knipperen, omdat de routine in IRQ ligt. Om het programma zo kort mogelijk te houden is de knippering niet opgeheven. Als een van beide toetsen is ingedrukt, wordt de waarde voor de kleur van kader en achtergrond met 1 verhoogd. IRQ-routines worden 50-60 maal per seconde aangesproken, dus bij een ingedrukte toets wordt de kleur 50-60 per seconde maal verhoogd. Druk voor de juiste kleurencombinatie verschillende keren op de toets. Wilt u de routine in een ander programma gebruiken, laad en start hem dan voor het andere programma. Bij gelijktijdig gebruik van programma's kunnen er in een aantal gevallen problemen optreden: (1) als het tweede programma te lang is, (2) als het geheugenbereik vanaf 40704 dat deze routine gebruikt, op een andere manier bezet is, (3) als F1 en F3 in het andere programma een functie vervullen. De BASIC loader ziet er als volgt uit:

```
 10 FOR I=40704 TO 40745
 20 READ A
 30 POKE I,A
 40 S=S+A
 50 NEXT I
 60 IF S<>4625 THEN PRINT "FOUT IN DATA": END
 70 PRINT "DATA O.K."
 80 SYS 40704
 90 DATA 120,169, 15,141, 20,  3,169,159,141
100 DATA  21,  3,133, 56, 88, 96, 72,165,203
110 DATA 201,  4,240,  8,201,  5,240, 10,104
120 DATA  76, 49,234,238, 32,208, 76, 26,159
130 DATA 238, 33,208, 76, 26,159
```

En dit is de de assemblerlisting:

```
9F00  SEI        : interrupt uit
9F01  LDA #$0F : veranderen van vectoren
9F03  STA $0314
9F06  LDA #$9F
9F08  STA $0315
9F0B  STA $38    : routine stoppen
```

```
9F0D CLI       : interrupt aan
9F0E RTS       : einde van deel 1
9F0F PHA       : register redden
9F10 LDA $CB   : uitlezen van toetsenbord
9F12 CMP #$04  : F1?
9F14 BEQ #9F1E
9F16 CMP #$05  : F2?
9F18 BEQ $9F24
9F1A PLA
9F1B JMP $EA31: zet IRQ voort
9F1E INC $D020: verhoogt kleur van kader
9F21 JMP $9F1A
9F24 INC $D021: verhoogt kleur van achtergrond
9F27 JMP $9F1A
```

Als in het programma, dat tegelijk met deze routine loopt, F1 en F3 al voorkomen, wijzig dan de waarden in regel 110. Daarvoor kunt u de decoderingstabel voor het toetsenbord in de bijlage raadplegen.

2.16 Twee beeldschermen

Met de routine in deze paragraaf kunt u twee beeldschermen gebruiken. Vooral voor grafische doeleinden (omschakeling van low res naar high res) is dat van belang. Uit didactische overwegigingen beperken we ons tot normale low res beeldschermen. Het voordeel bij spelletjes is, dat er al een tweede beeld kan worden opgebouwd, terwijl de speler nog met het eerste bezig is. Maar ook voor bijvoorbeeld tekstverwerking biedt een tweede scherm nieuwe mogelijkheden. Geef na het starten van het programma NEW (zie assemblerlisting), anders stopt de computer bij de invoer van een programma. Als u met F1 naar het tweede scherm bent gegaan kan het voorkomen, dat de cursor verdwenen is. Geef RETURN of een vergelijkbare instructie en hij verschijnt weer. Bij de eerste omschakeling staat het andere scherm vol met tekens, omdat dit bereik het domein was van het BASIC-geheugen. U kunt ze gewoon wissen. Omdat de schermen hetzelfde kleurgeheugen hebben, wordt bij het omschakelen de informatie over de kleuren overgenomen. In de routine wordt dit verhinderd doordat het kleur-RAM op de actuele kleur van de cursor wordt gezet. Het tweede beeldschermgeheugen ligt in het adresbereik 2048($0800)-3047($0BE7). De adressen van de wijzers voor de sprites op het tweede scherm liggen bij 3064-3071.

	normale toestand	eerste scherm	tweede scherm
begin beeldscherm-geheugen	1024 $0400	1024 $0400	2048 $0400
einde beeldscherm-geheugen	2023 $07E7	2023 $07E7	3047 $0BE7
wijzer voor sprites	2040 $07F8	2040 $07F8	3064 $0BF8
begin BASIC-geheugen	2048 $0800	3072 $0C00	3072 $0C00
kleur-geheugen	55296 $D800	55296 $D800	55296 $D800
toestand te bereiken	SYS	F3	F1
met	64738		

U kunt de routines ook bij spelletjes gebruiken:

a) Laat de speler F1 of F3 indrukken
b) POKE 203,4 om op het tweede scherm te komen
   POKE 203,5 om op het eerste scherm te komen

De BASIC-loader ziet er als volgt uit:

```
  0 REM twee beeldschermen
 10 FOR I=40702 TO 40789
 20 READ A
 30 POKE I,A
 40 S=S+A
 50 NEXT I
 60 IF S=11070 THEN PRINT "DATA O.K.":
    POKE 3072,0: SYS 40702: END
 70 PRINT "FOUT IN DATA"
 80 END
 90 DATA 169, 12,133, 44,120,169, 17,141, 20
100 DATA   3,169,159,141, 21,  3,133, 56, 88
```

```
110 DATA  96,165,203,201,  4,208, 13,169, 37
120 DATA 141, 24,208,169,  8,141,136,  2, 76
130 DATA  50,159,201,  5,208, 36,169, 21,141
140 DATA  24,208,169,  4,141,136,  2,169,216
150 DATA 133,252,160,  0,132,251,173,134,  2
160 DATA 145,251,200,240, 13,192,232,208,247
170 DATA 166,252,224,219,208,241, 76, 49,234
180 DATA 230,252, 76, 61,159
```

Het programma in machinetaal:

```
9EFE  LDA #$0C
9F00  STA $2C
9F02  SEI
9F03  LDA #$11
9F05  STA $0314
9F08  LDA #$9F
9F0A  STA $0315
9F0D  STA $38
9F0F  CLI
9F10  RTS
9F11  LDA $CB
9F13  CMP #$04
9F15  BNE $9F24
9F17  LDA #$25
9F19  STA $D018
9F1C  LDA #$08
9F1E  STA $0288
9F21  JMP $9F32
9F24  CMP #$05
9F26  BNE $9F4C
9F28  LDA #$15
9F2A  STA $D018
9F2D  LDA #$04
9F2F  STA $0288
9F32  LDA #$D8
9F34  STA $FC
9F36  LDY #$00
9F38  STY $FB
9F3A  LDA $0286
9F3D  STA ($FB),Y
9F3F  INY
9F40  BEQ $9F4F
9F42  CPY #$E8
9F44  BNE $9F3D
9F46  LDX $FC
9F48  CPX #$DB
9F4A  BNE $9F3D
9F4C  JMP $EA31
```

```
9F4F  INC $FC
9F51  JMP $9F3D
```

Save het beeld (hardcopy voor diskdrive of datasette) op de volgende manier:

a) Noteer de inhoud van de adressen 45 en 46
b) Geef de volgende POKE-instructie:

```
POKE 43,0: POKE 44,X
```

X is het scherm dat wordt gesaved: 4 is het eerste, 8 het tweede beeldscherm.

c) Ga verder met:

```
POKE 45,0: POKE 46,X+4
```

d) Lees SAVE "(naam)",8 in.
e) Zet na de SAVE-actie de BASIC-start terug met:

```
POKE 43,1: POKE 44,12: POKE 3072,0
POKE 45,...: POKE 46,...        (genoteerde waarden)
```

2.17 Lichtkrant in machinetaal

In hoofdstuk 6 staat een klein lichtkrantprogramma in BASIC. In deze paragraaf introduceren we zo'n programma in machinetaal. In tegenstelling tot de BASIC-routine loopt het machinetaalprogramma over IRQ. Het kan dus in de direkte modus en ook in BASIC programma's draaien en zelfs in enkele progamma's in machinetaal. De tekst verschijnt in de eerste regel van het beeldscherm en mag uit maximaal 40 letters bestaan. Bovendien accepteert de computer in eerste instantie alleen letters. Om grafische symbolen te kunnen gebruiken moet u de waarden daarvan direkt in de geheugenplaatsen poken. De tekst wordt opgeslagen vanaf 40448. De kleur van de tekst kan worden veranderd met

```
POKE 40760,(kleur)
```

De snelheid, waarmee de tekst beweegt, regelt u met

```
POKE 40783,(snelheid)
```

Gewoonlijk bevat dit adres de waarde 5. Snelheid en kleur kunnen ook in het programma worden veranderd. De kleurwaarde staat op de tweede plaats in DATA-regel 260, de waarde van de snelheid op de zevende plaats in regel 280. Ook de proefsom in regel 60 moet worden veranderd. In BASIC ziet het programma er als volgt uit:

```
  0 REM LICHTKRANT
 10 FOR I=40704 TO 40789
 20 READ A
 30 POKE I,A
 40 S=S+A
 50 NEXT I
 60 IF S<>10107 THEN PRINT "?FOUT IN DATA": END
 70 PRINT "DATA O.K."
 80 PRINT: POKE 19,1
 90 INPUT "tekst :"; A$
100 POKE 19,0: PRINT
110 FOR I=1 TO LEN (A$)
120 POKE 40448+I-1, ASC(MID$(A$,I,1)) AND NOT 64
130 NEXT I
140 FOR I=LEN(A$) TO 40
150 POKE 40448+I,32
160 NEXT I
170 SYS 40704
180 INPUT "NIEUWE TEKST (J-N)"; A$
190 IF A$= "J" THEN 80
200 DATA 120,169, 28,141, 20,  3,169,159,141
210 DATA  21,  3, 88,133, 56,162,  0,189,  0
220 DATA 158,157, 40,158,232,224, 40,144,245
230 DATA  96,206,255,159,208, 50,162,  0,173
240 DATA  50,159,105,  1,201, 40,144,  2,169
250 DATA   0,141, 50,159,189, 26,158,157,  0
260 DATA   4,169,  1,157,  0,216,232,224, 40
270 DATA 144,240,174, 50,159,232,224, 40,144
280 DATA   2,162,  0,142, 50,159,162,  5,142
290 DATA 255,159, 76, 49,234
```

Om dit programma tegelijk met een ander te laten lopen moet u het eerst in de computer laden en starten. Daarna wordt het tweede programma geladen, waarbij over de BASIC-loader heen wordt geschreven. Dat is niet erg, want de lichtkrant zit dan al in IRQ. In assembler ziet het er zo uit:

```
9F00  SEI
9F01  LDA #$1C
9F03  STA $0314
9F06  LDA #$9F
```

```
9F08  STA $0315
9F0B  CLI
9F0C  STA $38
9F0E  LDX #$00
9F10  LDA $9E00.X
9F13  STA $9F28,X
9F16  INX
9F17  CPX #$28
9F19  BCC $9F10
9F1B  RTS
9F1C  DEC $9FFF
9F1F  BNE $9F53
9F21  LDX #$00
9F23  LDA $9F32
9G26  ADC #$01
9F28  CMP #$28
9F2A  BCC $9F2E
9F2C  LDA #$00
9F2E  STA $9F32
9F31  LDA $9E1B,X
9F34  STA $0400,X
9F37  LDA #$01
9F39  STA $D800,X
9F3C  INX
9F3D  CPX #$28
9F3F  BCC $9F31
9F41  LDX $9F32
9F44  INX
9F45  CPX #$28
9F47  BCC $9F4B
9F49  LDX #$00
9F4B  STX $9F32
9F4E  LDX #$0553
9F50  STX $9FFF
9F53  JMP $EA31
```

De routine is opgebouwd volgens het schema in paragraaf 7.4.

2.18 De STOP-functie

De routine loopt over IRQ. Een druk op F1 stopt de computer tot u F3 gebruikt. In de direkte modus kan de routine als LIST-stop dienen, bijvoorbeeld om lange programma's te bestuderen. U hoeft

niet eerst op BREAK te drukken en daarna de LIST-instructie opnieuw in te voeren. En ook een BASIC-programma en enkele spelletjes in machinetaal kunt u ermee stopzetten. Het programma in BASIC:

```
  0 REM STOP-FUNCTIE
 10 FOR I=40704 TO 40752
 20 READ A
 30 POKE I,A
 40 S=S+A
 50 NEXT I
 60 IF S <> 5727 THEN PRINT "FOUT IN DATA " : END
 70 PRINT "DATA OK"
 80 SYS 40704
 90 DATA 120,169, 15,141, 20,  3,169,159,141
100 DATA  21,  3,133, 56, 88, 96, 72,165,203
110 DATA 201,  4,240,  4,104, 76, 49,234,169
120 DATA 159, 72,169, 41, 72,  8, 72,138, 72
130 DATA 152, 72, 76, 49,234,165,203,201,  5
140 DATA 208,235,240,229
```

De assemblerlisting:

```
9F00 SEI
9F01 LDA  #$0F
9F03 STA  $0314
9F06 LDA  #$9F
9F08 STA  $0315
9F0B STA  $38
9F0D CLI
9F0E RTS
9F0F PHA
9F10 LDA  $CB
9F12 CMP  #04
9F14 BEQ  $9F1A
9F16 PLA
9F17 JMP  $EA31
9F1A LDA  $9F
9F1C PHA
9F1D LDA  #29
9F1F PHA
9F20 PHP
9F21 PHA
9F22 TXA
9F23 PHA
9F24 TYA
```

```
9F25 PHA
9F26 JMP  $EA31
9F29 LDA  $CB
9F2B CMP  #05
9F2D BNE  $9F1A
9F2F BEQ  $9F16
```

Hebben F1 en F3 al een functie, gebruik dan andere toetsen door de waarden in de adressen $9F12 en $9F2B te veranderen. Als u geen machinetaalmonitor hebt en met de BASIC-lader werkt, moet de eerste 4 in regel 110 en de 5 in 130 gewijzigd worden. Raadpleeg de decodeertabel voor het toetsenbord in de bijlage.

2.19 Toeval? Opmerkingen over RND

De RND-routine kiest niet toevallig een getal, want de computer werkt logisch. Aan de keuze ligt een gecompliceerd algoritme ten grondslag:

1. De laatste RND-waarde (adressen 193-143) wordt in de accu voor reële getallen (FAC) geladen.

2. De FAC wordt vermenigvuldigd met constante 11879546 ($E08D-$E091).

3. Vervolgens wordt constante 3.92767774E-08 ($E092-$E096) bij het resultaat opgeteld.

4. Dan wordt er met de inhoud van enkele geheugenposities gemanipuleerd:

$65 en $62 verwisselen van plaats.
$63 en $64 verwisselen van plaats.
In $66 komt 0 te staan.
In $61 komt $80 te staan.

5. Tenslotte krijgt de FAC aan de linkerkant een rechte kantlijn, wordt afgerond en opgeslagen.

In het bereik, waar de constanten liggen, staan de volgende hexadecimale getallen:

```
$E08D  98 35 44 7A 00
$E092  68 28 B1 46 00
```

Als u niets over de FAC weet, wordt het moeilijk om uit die getallen de twee constanten te destilleren. Het volgende programma doet dat voor u:

```
LDA #$LB
LDY #$HB
JSR $BBA2   ; GETAL IN FAC
JSR $BDDD   ; FAC IN ASCII VERANDEREN
              EN VANAF $0100 OPSLAAN
LDA #$00    ; LOW-BYTE ADRES VAN STRING
LDY #$01    ; HIGH-BYTE ADRES VAN STRING
JSR $AB1E   ; STRING UITVOEREN
RTS
```

LSB en MSB geven het adres aan, waar vanaf de (5) bytes staan, die in de FAC moeten komen. Wilt u bekijken wat er in de FAC staat, geef dan adres $0061. In BASIC ziet het programma (in de cassettebuffer) er als volgt uit:

```
10 FOR X=0 TO 17: READ A: POKE 828+X,A: NEXT X
20 DATA 169, LB,160, HB, 32,162,187
30 DATA  32,221,189,169,  0,160,  1
40 DATA  32, 30,171, 96
```

Start de routine met SYS 828 en substitueer de waarden voor LSB en MSB. Die kunt u ook POKEn:

```
POKE 829,LSB: POKE 831,MSB
```

Voorbeeld:

LSB	MSB	WAARDE
141	224	constante 1
146	224	constante 2
139	0	laatste RND-waarde
97	0	FAC

U kunt ook in machinetaal naar de RND-routine toespringen (sprongadres $E0BE (57534)). Een equivalent voor RND(-1) en RND(0) hebben wij niet kunnen vinden, maar probeer eens andere sprongadressen tussen $E097 en $E0EF. Door een teller te lezen kan ook in machinetaal een toevallig getal worden gemaakt. Denk bijvoorbeeld aan adres 160-162 (klok) en 53266 (de laatste regel,

die de elektronenstraal op het scherm schreef). Niet alleen voor spelletjes kan RND goed worden gebruikt, maar ook om files te beveiligen, die niet voor iedereen bedoeld zijn. De laatste RND-waarde ligt in 139-143. Omdat de computer een nieuwe waarde altijd met behulp van de laatste berekent, kunt u voor RND een vaste beginwaarde kiezen. Bij elke doorloop worden de RND-waarden altijd meteen berekend. Getallen, die op grond van RND tot stand zijn gekomen, kunnen dus worden gecodeerd en bij een nieuwe doorloop weer gedecodeerd. In het volgende programma is dit principe nader uitgewerkt.

```
10 GOSUB 1000
20 PRINT CHR$(147): X=20480
30 POKE 204,0
40 GETA$: IF A$="" THEN 40
50 IF A$=CHR$(133) THEN 100
60 IF ASC(A$) <32 OR ASC(A$)>95 THEN 40
70 IF PEEK(207) THEN 70
80 POKE 204,1: PRINT A$;
90 FOR YY=0 TO 3: Y=Y+INT(RND(1)*40):NEXT: Y=Y+ASC(A$)
95 POKE X,Y: X=X+1: Y=0: GOTO 30
100 POKE X,0: GOSUB 1000
110 PRINT CHR$(147): X=20480
120 Z=PEEK(X): IF Z=0 THEN END
130 FOR YY=0 TO 3: Z=Z-INT(RND(1)*40):NEXTYY
140 PRINT CHR$(Z);:X=X+1: Z=0: GOTO 120
1000 PRINT CHR$(147): FOR X=0 TO 4
1010 INPUT A:IF A<0 OR A>255 THEN 1010
1020 POKE 139+X,A
1030 NEXT X
1040 RETURN
```

Start het programma en toets vijf waarden in, nu kunt u uw tekst intoetsen. Druk op F1 als u klaar bent. Het intoetsen van dezelfde vijf waarden is voor de computer aanleiding om de tekst weer op het scherm te laten verschijnen. Is de tweede set waarden waarden niet gelijk aan de eerste vijf, dan wordt de tekst min of meer verminkt. Het programma is niet absoluut veilig, omdat met kleine afwijkingen in de waarden geen rekening wordt gehouden. Dat ligt aan de INT-functie in regel 90 en 130. Maar het is nu wel duidelijk hoe gegevens kunnen worden gecodeerd:

a. Kies de RND-waarde door bepaalde waarden (code van gebruiker) in de geheugenplaatsen 139-143 te POKEn.

b. Codeer de gegevens met de RND-functie. U bepaalt zelf de manier, waarop.

c. Voer bij het inlezen van de gegevens een sleutel in, waarmee RND weer op een bepaalde waarde wordt gezet.

d. Met RND worden de gegevens gedecodeerd. Is de verkeerde sleutel gebruikt, dan worden de gegevens verkeerd uitgevoerd.

Theoretisch kan iemand die alle mogelijke waarden voor de gebruikerscode probeert, toegang krijgen tot de gegevens. Maar in de praktijk duurt dat erg lang. Als extra maatregel kunt u ook de functie van de sleutel (bijvoorbeeld Y=Y+RND(1)) in de beveiliging opnemen. De combinatiemogelijkheden zijn nu oneindig groot.

Met de instructie RND kan in BASIC een schijnbaar toevallig getal worden gecreëerd. U kunt zelf kiezen of het argument, dat noodzakelijk volgt, positief, negatief of nul is. Een positief argument produceert een schijnbaar toevallig getal op grond van de laatste random waarde. Bij een negatief argument is de basis (adres 193-143) afhankelijk van de waarde van het negatieve argument. Om een werkelijk toevallig getal te krijgen moet de basis opnieuw worden bepaald door in het programma regel 10 te vervangen:

```
10 X=RND(-TI): X=INT(RND(1)*...
```

2.20 INPUT-variaties

De Commodore 64 kent alleen de invoerinstructies GET en INPUT. Voor een hoop toepassingen is dat niet voldoende. Bij GET ontbreekt de cursor en bij INPUT levert het vraagteken problemen op. Bovendien accepteert INPUT sommige leestekens niet, omdat ze een functie vervullen. De standaard INPUT-instructie is een vraag en geen bevel. Om het een bevel te laten zijn moet het vraagteken worden uitgeschakeld. Dat kan met:

```
10 POKE 19,1: INPUT "<RETURN>";A$: PRINT: POKE 19,0
```

of

```
10 OPEN 1,0
20 INPUT#1,A$
30 CLOSE 1
```

U opent dus gewoon het toetsenbord. Om ook leestekens in te kunnen voeren, bijvoorbeeld bij teksten, hebben we een BASIC-programma ontworpen. Het is snel genoeg voor tekstverwerking. In vergelijking met de normale INPUT zijn er de volgende voordelen:

a) Een string kan uit een willekeurig aantal elementen bestaan. Daarvoor hoeft alleen de DIM-instructie te worden veranderd.

b) Ook stuurtekens, die bij de uitvoer van de string in acht worden genomen, kunnen worden ingelezen.

c) Er kan alleen tekst worden ingevoerd. Die wordt in IN$ opgeslagen. Met VAL kunnen ook getallen worden ingevoerd.

```
10 REM GESIMULEERDE INPUT
20 FR$="HOE LUIDT DE INVOER ??":GOSUB60000:
   REM OPROEP ROUTINE
25 PRINTIN$
30 END
40 :
50 :
60 :
60000 REM GESIMULEERDE INPUT (ROUTINE)
60010 DIM IN$(1000):PRINT FR$;" ";
60050 POKE 204,0
60051 POKE 212,1:GET IN$:IF IN$=""THEN 60051
60052 POKE 207,0
60055 IF IN$=CHR$(13) THEN POKE 204,1:PRINT" ":
      IN$="":GOTO 60100:REM RETURN-TOETS
60056 IF IN$=CHR$(20) THEN GOTO 60120
60060 PRINT IN$;
60070 IN$(IN)=IN$:IN=IN+1
60080 GOTO 60050
60100 FOR SL=0 TO IN-1
60101 IN$=IN$+IN$(SL)
60102 NEXT:RETURN
60120 IF IN=0 THEN GOTO 60050
60125 POKE 212,0:POKE 207,0:PRINT CHR$(157)
      CHR$(32)CHR$(32)CHR$(157)CHR$(157);
60130 IN=IN-1:IN$(IN)="":GOTO 60050
```

Belangrijke adressen:

212 - flag voor aanhalingsmodus
204 - cursor in- en uitschakelen

Belangrijke variabelen:

IN$ - resultaat van het uitlezen
FR$ - vraagstring (bij oproep van routine op scherm)

Ook met

```
10 POKE 198,1: POKE 631,34: INPUT A$
```

kunt u in een INPUT-instructie een komma gebruiken. Binnen aanhalingstekens vervullen komma's geen andere functie dan die van komma. Het laden van een aanhalingsteken (code 34) in de toetsenbordbuffer (adres 631-640), wordt via adres 198 aan de computer doorgegeven. Het aanhalingsteken, dat bij de aansluitende INPUT-instructie wordt uitgevoerd betekent, dat er komma's kunnen worden ingelezen. Dit aanhalingsteken speelt geen rol in de string en beïnvloedt evenmin de LEN-instructie.

2.21 Trucs met de diskdrive

Op grond van de hardware kunnen er maximaal 8 diskdives op de Commodore 64 worden aangesloten (kanalen 8-15). Het nummer van de diskdrive (8) moet worden veranderd. Dat doet het volgende programma:

```
10 INPUT"OUDE NUMMER";ON
20 INPUT"NIEUWE NUMMER";NN
30 OPEN 1,ON,15
40 PRINT#1,"M-W"CHR$(119);CHR$(0);
   CHR$(2);CHR$(NN+32);CHR$(NN+64)
50 CLOSE 1
```

Sluit een diskdrive aan en kies als adres van het apparaat een ander getal dan 8. Omdat de hardware bepaalt, dat een tweede diskdrive adres 8 heeft, kunnen ze allebei eenduidig worden aangesproken. Wilt u een derde aansluiten, verander dan het adres van de tweede, enzovoort. Met het volgende programma heeft u daar geen omkijken naar:

```
10 PRINT"HOEVEEL DISKDRIVES WILT U AANSLUITEN"
20 INPUT AY
30 IF AY<1 OR AY>8 THEN 20
40 AN=8; FOR Y=1 TO AY
```

```
 50 PRINT"SCHAKEL NU DRIVE";Y;" IN"
 60 GET A$: IF A$="" THEN 60
 70 OPEN 1,AN,15
 80 PRINT#1,"M-W"CHR$(119);CHR$(0);
    CHR$(2);CHR$(39+Y);CHR$(71+Y)
 90 CLOSE1: PRINT" DEZE DRIVE HEEFT NU NUMMER"AN
100 AN=AN+1: NEXT Y
```

Schakelt u de diskdrive uit, dan wordt zijn adres weer 8. Dit nadeel kan worden voorkomen door het primaire adres via de hardware te veranderen. Omdat bij die methode alleen de nummers 8 tot 11 kunnen worden gekozen, moet u bij meer dan 4 diskdrives uw toevlucht nemen tot de software-oplossing. Het initialiseren van de diskdrives, dat dan elke keer moet gebeuren, gaat het beste met een EPROM, gekoppeld aan het programma en een autostart. Maar u kunt natuurlijk ook het tweede programma steeds opnieuw laden.

Omdat bij de hardware-oplossing de diskdrive moet worden geopend, vervalt de garantie. Geen zorgen: als u voorzichtig bent, gebeurt er niets. Trek eerst de stekker uit het apparaat en draai de 4 schroeven aan de onderkant voorzichtig los. Door te hard schudden kan de schrijf-leeskop van de diskdrive kapot gaan. Als u de kap heeft verwijderd, ziet u midden op de printplaat een aantal flinke IC's. Onder de CPU, een 6502, bevindt zich een 6522 voor in- en uitvoer. Als u van daaruit naar de voorkant van de diskdrive gaat, komt u bij C64, een elektrolytische condensator. Daarnaast bevinden zich twee soldeerbruggen in de vorm van halve cirkels, die door een smalle verbinding aan elkaar zijn gekoppeld. Door een brug of beide bruggen door te knippen bepaalt u het nieuwe primaire adres van de diskdrive:

onderbroken halve cirkel	nummer van apparaat
geen	8
1 (rechts)	9
2 (links)	10
1 en 2	11

Het nummer van de diskdrive blijft dus ook na het uitschakelen gelijk.

Gewoonlijk kan een eenvoudige diskette aan twee kanten worden gebruikt door een nieuwe inkeping te maken. Maar het kan ook anders. Op een van de lange kanten van de printplaat bevinden zich talrijke aansluitingen. Tussen de oranje en de violette kabel van de grootste (P6) plaatst u een schakelaar buiten de diskdrive. De diskettes kunnen dan zonder inkeping aan twee kanten worden gebruikt, omdat de lichtbundel, die de inkeping

controleert, door een gesloten schakelaar wordt overbrugd. Savet u in een programma bestanden en u hebt vergeten de diskdrive in te schakelen, dan hangt het programma zich op met de foutmelding "DEVICE NOT PRESENT". Het zou beter zijn, dat de gebruiker erop attent werd gemaakt dat zijn diskdrive niet aan staat. De volgende routine, die bekijkt of dat het geval is, zou deel moeten uitmaken van elk programma, dat met de diskdrive werkt. Is de diskdrive ingeschakeld, dan wordt geheugenplaats $FF(255) op 0 gezet, zo niet op 5. Uw programma kan de waarde opvragen en erop reageren. De routine ligt in de cassettebuffer, die niet gebruikt wordt als u met een diskdrive werkt.

```
033C LDA #$01  :LENGTE FILENAAM
033E LDX #$D0  :ADRES LOW
0340 LDY #$FF  :ADRES HIGH
0342 JSR $FFBD :FILENAAMPARAMETER ZETTEN
0345 LDA #$01  :LOGISCH FILENUMMER
0347 LDX #$08  :DEVICENUMMER
0349 LDY #$00  :SEC. ADRES
034B JSR $FFBA :FILEPARAMETER ZETTEN
034E JSR $FFC0 :FILE OPENEN(OPEN)
0351 BCS $0356 :GEEN REACTIE VAN DEVICE
0353 LDA #$00  :DEVICE AANWEZIG
0355 STA $FF   :FLAG OPSLAAN
0357 LDA #$01  :LOGISCH FILENUMMER
0359 JSR $FFC3 :FILE SLUITEN(CLOSE)
035C RTS
```

Als adres van de filenaam is $FFD0 gekozen, omdat daarin de waarde 36 staat (ASCII-waarde voor $). $FFE5 (filenaam *) had tot "FILE NOT FOUND" geleid, omdat natuurlijk niet een willekeurige naam kan worden gezocht. Het programma in BASIC:

```
10 FOR Y=0 TO 32: READ A:POKE 828+Y,A:NEXTY
20 DATA 169,  1,162,208,160,255, 32,189,255
30 DATA 169,  1,162,  8,160,  0, 32,186,255
40 DATA  32,192,255,176,  3,169,  0,133,255
50 DATA 169,  1, 32,195,255, 96
```

Met SYS 828 roept u deze routine op. Het volgende programma in deze paragraaf leest het foutkanaal van de diskdrive. Hoe dat in BASIC kan stond in paragraaf 2.5. Van die routine gaan we machinetaal maken. Wie de KERNAL kent (hoofdstuk 10) weet, dat er een speciale OPEN-routine is. Bij INPUT#1 wordt het al moeilijker, maar de KERNAL heeft een routine die een teken van de IEC-bus kan halen. In het onderstaande machinetaalprogramma speelt de OPEN-routine geen rol. Om in machinetaal het foutkanaal van de diskdrive te kunnen uitlezen hoeft het BASIC-programma in

paragraaf 2.5 niet slaafs te worden gevolgd. Het machinetaalprogramma bevindt zich in de cassettebuffer.

```
033C LDA #$08  :DEVICENUMMER
033E STA $BA   :OPSLAAN
0340 JSR $FFB4 :TALK ZENDEN(FLOPPY ZENDT)
0343 LDA #$6F  :SEC. ADRES
0345 STA $B9   :OPSLAAN
0347 JSR $FF96 :SEC. ADRES NAAR TALK
034A JSR $FFA5 :TEKEN VAN FLOPPY HALEN
034E JSR $FFD2 :TEKEN UITVOEREN
0351 CMP #$0D  :RETURN?
0353 BNE $034B :NEE, DAN VERDER
0355 JSR $FFAB :UNTALK ZENDEN(STOP MET ZENDEN)
0357 RTS       :TERUG
```

Het secundaire adres 15 van de BASIC-variant ligt in adres $0F. $60 is voor de diskdrive. Meestal wordt $60 in de OPEN-routine bij het secundaire adres opgeteld, maar dat moet in dit geval meteen gebeuren, omdat er geen OPEN-routine is. In BASIC vertaald ziet de machinetaalroutine er zo uit:

```
10 FOR X= 0 TO 27: READ A:POKE 828+X,A: NEXT X
20 DATA 169,  8,133,186, 32,180,255
30 DATA 169,111,133,185, 32,150,255
30 DATA  32,165,255, 32,210,255,201
40 DATA  13,208,246, 32,171,255, 96
```

Met SYS 828 roept u het programma op.

Tot slot enkele tips. Het is ook mogelijk bepaalde delen van de directory te laden:

```
LOAD "$0:(naam)",8
```

Staat het programma niet op deze diskette, dan wordt alleen de diskheader (naam van disk) gegeven. U kunt ook de ster gebruiken:

```
LOAD "$0:PAS*",8
```

Alle programma's, die met PAS beginnen, worden opgevoerd. Door SHIFT en RUN/STOP tegelijk in te drukken kunnen programma's op

band worden geladen en automatisch gestart. Dat kan ook met diskettes:

```
LOAD "GRAFIEK",8
```

Na de normale LOAD-invoer drukt u in plaats van RETURN tegelijk SHIFT en RUN/STOP in. Op het scherm verschijnt "LOAD", het programma wordt geladen en automatisch gestart. Ook in dit geval kan de ster worden gebruikt:

```
LOAD "*",8        (SHIFT en RUN/STOP)
```

Het eerste programma op de diskette wordt geladen en gestart. Het laden van de directory kunt u ook beperken tot bepaalde programmatypes:

```
LOAD "$*=T",8
```

Kies T op deze manier:

T	TYPE FILE
P	programma
S	sequentieel
R	relatief
U	user

De codes kunnen voor alle diskdrive-instructies worden gebruikt. U kunt bijvoorbeeld alleen userfiles wissen:

```
OPEN 15,8,15
PRINT#1,"S:*=U"
```

3 BEVEILIGING VAN SOFTWARE

3.1 De LIST-functie

LIST laat de programmaregels op het scherm verschijnen. Wacht- en codewoorden kunnen worden gelezen en het programma kan worden veranderd. In de volgende paragrafen behandelen we onder andere hoe het listen kan worden verhinderd.

3.2 Listen zonder regelnummers

Door de normale waarde (25) van adres 22 (wijzer op actuele stringstack) te veranderen in 35 verschijnen er bij het listen geen regelnummers.

```
POKE 22,35
```

zorgt daarvoor. Met

```
POKE 22,25
```

komt u in de oude toestand terug, net als na elke foutmelding. Zulke kleine veranderingen hebben altijd neveneffecten. In het geval van POKE 22,35 wordt geen enkele PRINT-instructie uitgevoerd als u het programma runt. We hebben de waarde van adres 22 verschillende malen veranderd met het volgende resultaat:

```
POKE 22,25 - normale toestand
POKE 22,32 - regelnummers worden onleesbaar
POKE 22,33 - geen regelnummers, maar uitroeptekens
POKE 22,34 - ?FORMULA TOO COMPLEX ERROR, normale toestand
POKE 22,35 - regelnummers verdwijnen
```

De laatste regel, die bij het printen van een programma verdwijnt, verschijnt pas na

```
PRINT#1   (of gebruikte filenummer)
```

3.3 LIST-beveiliging - LIST-functie uitschakelen

De volgende REM-instructie zorgt ervoor, dat de programmaregels niet meer worden gelist. In plaats daarvan verschijnt SYNTAX ERROR op het scherm:

```
10 PRINT "LISTBEVEILIGINGSVARIANT 1"
20 REM druk SHIFT en L tegelijk in
30 PRINT "LISTEN ONMOGELIJK"
```

Deze truc is gemakkelijk te doorzien, tenzij u hem met een andere beveiliging combineert, bijvoorbeeld met het manipuleren met regelnummers. Een beveiliging, die bestaat uit om de 5 regels zo'n REM-instructie, is praktisch niet te kraken.

De tweede variant ziet er op het eerste gezicht helemaal niet als een LIST-beveiliging uit. Lees het volgende programma in:

```
10 REM LIST-beveiligingsvariant 2
20 PRINT "BEVEILIGD": REM""
```

Breng de cursor de eerste aanhalingstekens achter REM in regel 20 en tik net zovaak INST als het aantal voorafgaande posities in die regel (26). Druk vervolgens 26 keer DEL. Achter de REM-instructie in regel 20 staan nu 26 reverse T's. Elke reverse T wist een teken van deze regel. Bij het listen wordt de uitgevoerde beveiligde regel direkt door de reverse T's gewist. Dit werkt alleen bij regels kleiner dan 10 karakters, de REM-instructie niet meegerekend. Langere regels verraden zichzelf door op te lichten.

De volgende routine achter een programma is een beproefd middel tegen krakers:

```
62000 FOR A=PEEK(43)+256*PEEK(44) TO PEEK(45)+256*PEEK(46)-3
62010 IF PEEK(A)=58 AND PEEK(A+1)=58 AND PEEK(A+2)=58 THEN
      GOSUB 62030
62020 NEXT A: END
62030 IF PEEK(A+3)=58 ANDPEEK(A+4)=58 THEN POKE A,0: A=A+4:
      RETURN
```

De te beveiligen regels worden door vijf dubbele punten gekenmerkt.

```
45 PRINT "VOORBEELD"
```

wordt

```
45 ::::PRINT "VOORBEELD"
```

Start de beveiligingsroutine met RUN 62000. Hoe langer het programma dat moet worden beveiligd, des te later verschijnt de cursor. Probeer de beveiliging eerst met kleinere programma's om vertrouwd te raken met de snelheid van het proces. Als het programma wordt gelist, verschijnen de nummers van de beveiligde regels, maar niet de inhoud. Wis vervolgens de regels 62000-62030 en sla het beveiligde programma op.

Het principe van de beveiliging is, dat eerst een lus wordt geïnitialiseerd, die het hele BASIC-programma in het geheugen doorloopt (43-44 (BASIC-start), 45-46 (einde programma)). Zijn de vijf dubbele punten (code 58) gevonden, dan wordt alleen de eerste gewijzigd en wel in code 0. Voor de LIST-routine betekent die code het einde van de regels: het listen houdt op. De uitvoering van de instructie loopt geen gevaar, omdat de vier dubbele punten als koppeltekens worden geïnterpreteerd. Het volgende machinetaalprogramma is vergelijkbaar, maar veel sneller: een routine van 13 Kbyte beveiligt het in ongeveer 1 seconde. In BASIC ziet het er zo uit:

```
10 REM MACHINETAALPRG OM BASIC-LISTINGS
TE BEVEILIGEN
20 REM BEGIN $8000 (DEC.32768), LENGTE:
80 BYTES
30 REM BEVEILIGDE REGEL MOET MET > EN 4
DUBBELE PUNTEN GEMARKEERD WORDEN:
40 REM VOORBEELD: I.P.V.'10 REM' LUIDT D
E REGEL NU: '10 >::::REM'
50 REM M-PRG STARTEN DOOR SYS8*4096
60 :
70 :
80 :
90 READA:REM M-PRG INLEZEN
100 IFA=-1THENGOTO150:REM EINDE-MARKERIN
G CONTROLEREN
110 CH=CH+A:REM BEREKENING VAN PROEF-SOM
120 POKE8*4096+I,A:REM OPSLAAN VAN GEGEV
ENS
130 I=I+1:GOTO70:REM VOLGENDE WAARDE HAL
EN
150 IFCH<>9839THENPRINT"FOUT IN DATA!":L
```

```
IST200-
160 PRINT"DATA OK.  STARTEN V/H PRG DOO
R    SYS 8*4096 !":END
200 DATA162,120,169,8,160,0,132,34,133,3
5,177,34,201,177,240,9,200,208
210 DATA247,230,35,202,208,242,96,142,80
,128,141,81,128,140,82,128,165,35
220 DATA133,37,165,34,133,36,162,4,200,2
40,28,177,36,201,58,208,10,202
230 DATA208,244,172,82,128,169,0,145,34,
174,80,128,173,81,128,172,82,128,76
240 DATA16,128,230,37,76,47,128,-1
```

Met SYS 8*4096 roept u de routine op. De regels die moeten worden beveiligd zijn niet met vijf dubbele punten, maar met >:::: gemarkeerd.

3.4 Veranderen van de BASIC-LINK

Door BASIC-LINKS te veranderen kunnen listings ook worden beveiligd. Een gedeelte uit het BASIC-geheugen verduidelijkt het begrip BASIC-LINK.

```
2048    0
2049   16 LINK low
2050    8 LINK high
2051   10 REGELNUMMER low
2052    0 REGELNUMMER high
2053  153 begin van regelinhoud

2063    0 eind van regel
2064   31 LINK low (vorige LINK wijst op dit adres)
2065    8 LINK high
2066   20 REGELNUMMER low
2067    0 REGELNUMMER high
2068  153 begin van regelinhoud (153 = code voor PRINT)
```

De 0 in adres 2048 markeert de BASIC-start. Daarna komt de LINK. LSB en MSB in 2049 en 2050 vormen het beginadres van de volgende BASIC-regel. De bytes in 2051 en 2052 zijn het regelnummer van de actuele regel. Dan volgt de regelinhoud, een 0 markeert het einde. De LINK in 2064 bevat het adres van de volgende regel, waarop de vorige LINK wees. Daarna komt het nummer van die regel. Als er geen BASIC-regels meer volgen, is de laatste LINK 0,0. De LINK is alleen voor de LIST-routine echt belangrijk. Wilt

u een regel laten verdwijnen, dan hoeft u alleen de LINK op deze regel te veranderen door hem naar de volgende te laten verwijzen. Het volgende BASIC-programma doet dat automatisch:

```
60000 POKE53280,1:POKE53281,1:POKE646,6:
PRINT"■";
60001 PRINTTAB(10)"■ LINK-CHANGER "
60002 PRINTTAB(5)"■■■■■DIT PROGRAMMA MANI
PULEERT DE":PRINT
60003 PRINTTAB(5)"REGEL-LINKS ER WORDT N
U NAAR HET":PRINT
60004 PRINTTAB(5)"EERSTE EN HET TWEEDE R
EGELNUMMER":PRINT
60005 PRINTTAB(5)"GEVRAAGD":PRINT
60006 PRINTTAB(5)"ALLE REGELS TUSSEN DEZ
E NUMMERS":PRINT
60007 PRINTTAB(5)"INCL. HET TWEEDE NUMME
R VERDWIJNEN":PRINT
60008 PRINTTAB(5)"HOE LANGER UW PROGRAMM
A IS,":PRINT
60009 PRINTTAB(5)"DES TE LANGER DUURT HE
T OMZETTEN!":PRINT:PRINT
60010 PRINTTAB(10)"DRUK EEN TOETS!"
60011 GETA$: IFA$=""THEN 60011
60012 PRINT"■";
60015 INPUT"EERSTE REGELNUMMER";A:INPUT"
TWEEDE REGELNUMMMER";E
60019 PRINT"GEVONDEN REGELNUMMER:"
60020 Q=PEEK(43)+256*PEEK(44)
60030 ZL=PEEK(Q+2)+256*PEEK(Q+3):IFZL=AT
HENA1=Q:A2=Q+1:GOTO60060
60031 IF ZL>ATHENPRINT"■REGEL BESTAAT NI
ET !":END
60035 PRINTTAB(23)"■";ZL
60040 Q=PEEK(Q)+256*PEEK(Q+1):IFQ>=PEEK(
45)+256*PEEK(46)-3THENEND
60050 GOTO60030
60060 Q=PEEK(43)+256*PEEK(44)
60070 IFPEEK(Q+2)+256*PEEK(Q+3)=ETHENPOK
EA1,PEEK(Q):POKEA2,PEEK(Q+1):END
60080 Q=PEEK(Q)+256*PEEK(Q+1):IFQ>=PEEK(
45)+256*PEEK(46)-3THENEND
60090 GOTO60070
```

3.5 Regels wissen? SYNTAX ERROR!

Een andere methode om listings te beveiligen werkt met de regelnummers, die gesplitst in MSB's en LSB's in het BASIC-geheugen staan. Gewoonlijk leidt een nummer groter dan 63999 tot SYNTAX ERROR. In deze paragraaf wordt uitgelegd hoe die waarde tot 65535 kan worden verhoogd. Verder laten we zien hoe u een listing kunt laten bestaan uit verschillende regels met hetzelfde nummer of uit regels, die door elkaar zijn gegooid. Omdat een regelnummer bestaat uit MSB en LSB, is het kleinst mogelijke nummer 0 (0+256*0) en het grootste 65535 (255+256*255). De computer werkt programmaregels af, zoals ze in het geheugen staan. Een verandering van de nummers heeft daarom geen invloed op het verloop van het programma. Zorg er wel voor, dat de instructies GOTO, GOSUB en THEN aan de nieuwe situatie worden aangepast. Zelf het adres van het LSB en MSB van een regelnummer in het geheugen opzoeken duurt een eeuwigheid. Het volgende programma doet dat automatisch, het verandert moeiteloos regelnummers van lange programma's.

```
60000 Q=PEEK(43)+256*PEEK(44)
60001 PRINT"♥":GOSUB60100
60002 Q=PEEK(Q)+256*PEEK(Q+1):IFQ>=PEEK(
45)+256*PEEK(46)-3THENEND
60003 GOTO60001
60010 IFPEEK(Q)=0THENGOSUB60100
60020 NEXTQ:END
60100 LO=PEEK(Q+2):HI=PEEK(Q+3)
60105 POKE199,1:PRINT"REGELNUMMER-CHANGE
R":PRINT
60110 PRINT"GEVONDEN REGELNUMMER :";LO+2
56*HI
60120 PRINT:PRINT"[1] VERANDEREN"
60130 PRINT:PRINT"[2] VERDER"
60140 PRINT:PRINT"[3] EINDE"
60150 GETA$:IFA$=""THEN60150
60160 IFA$="2"THENRETURN
60170 IFA$="3"THENEND
60180 A$="0":PRINT:INPUT"NIEUW REGELNUMM
ER (0-65535)";A$:A=VAL(A$)
60190 HI=INT(A/256):LO=A-(256*HI)
60200 POKEQ+2,LO:POKEQ+3,HI:RETURN
```

Als in regel 60000 het eerste LINK-adres is bepaald, komt vanaf regel 60020 de routine in het spel. LSB en MSB van het eerste regelnummer worden opgevraagd (LINK-adres +2 en +3) en het nummer, dat daaruit resulteert, wordt als gevonden gemeld (60030). De keus is aan u: (1) veranderen of (2) verder gaan. (2) laat het regelnummer intact en zoekt het volgende. Bij (1) kan het nummer worden veranderd in een getal tussen 0 en 65535. Het decimale getal dat u inleest, wordt veranderd in LSB en MSB (60070) en als zodanig in het BASIC-geheugen gepoket. In regel 60010 wordt de oude LINK door de nieuwe vervangen en begint het proces van voren af aan. Als het programma niet wordt gestopt, gaat het door tot het einde van het programma in het geheugen bereikt is (regel 60010).

De voordelen van de gemanipuleerde regels zijn: een regel groter dan 63999 kan niet worden gewist door er overheen te schrijven. Is het nummer van een regel kleiner dan de vorige, dan kan de regel ook niet worden gewist. Dat kan alleen als die regel de eerste is van het programma. Is het programma veranderd, dan kunt u het hulpprogramma wissen en het hoofdprogramma opslaan.

3.6 Kunstmatige stuurtekens

De gewone stuurtekens zijn die van de cursor, die vaak in PRINT-instructies staan of die voor de kleur. Lees eens

```
10 REM" "
```

in en ga met de cursor naar het tweede aanhalingsteken. Druk dan tegelijk op CTRL en RVS ON/9 en vervolgens tegelijkertijd op SHIFT en M. Als kunstmatig stuurteken verschijnt een donker hokje met een lichte schuine streep. Nadat u tegelijk op SHIFT en Q hebt gedrukt, verschijnt het stuurteken CURSOR UP. Schakel de reverse modus uit door tegelijk CTRL en RVS OFF/0 in te drukken. Daarachter schrijft u TESTREGEL en u drukt op RETURN. Op het scherm staat

```
10 REM"▧⊤TESTREGEL
```

Als u dit LIST, wordt de regel onzichtbaar behalve TESTREGEL. Het eerste stuurteken zorgt ervoor, dat normale stuurtekens (in dit geval CURSOR UP) achter de REM-instructie worden uitgevoerd, maar ook voor een carriage return (CR). TESTREGEL bedekt dan 10 REM". Op de volgende manier maakt u er een LIST-beveiliging van:

```
100 REM"▒]90 SYS 4096
```

Het lijkt alsof in regel 90 een machinetaalprogramma wordt geactiveerd. Maak de regels na 100 met een van de voorafgáande methodes onzichtbaar en een buitenstaander ziet de routine aan voor een programma in machinetaal. het zal niet in hem opkomen, dat het om een verborgen BASIC-listing gaat. Het kan ook zo:

```
100 PRINT"BEVEILIGD!":REM"▒]
```

De regel wordt geLIST, maar direkt weer bedekt door de volgende.

3.7 Beveiliging met POKEs

Deze LIST-beveiliging (adressen 774 ($0306) en 774 ($0307) beïnvloedt de hele LIST-routine. Die adressen vormen de LIST-vector, die gewoonlijk naar $A71A in de BASIC-interpreter wijst, het beginadres van de routine, die BASIC-tokens vertaalt. De LIST-functie gebruikt deze routine. Als de LIST-vector naar een andere routine wijst, verandert de LIST-functie. Bijvoorbeeld:

```
POKE 774,226: POKE 775,252
```

Hierna wordt de computer met LIST gereset, omdat de POKEs de LIST-vector naar de RESET-routine ($EA31) laten wijzen. U kunt ook de vector op eigen machinetaalprogramma's zetten. Het nadeel van deze variant is, dat de beveiliging pas in werking treedt na het starten van de routine. Maar als u een autostart gebruikt, is deze optie uiterst interessant.

3.8 Het blokkeren van gevaarlijke toetsen

Als u de RUN/STOP-toets onschadelijk maakt, kan een programma tussentijds niet worden onderbroken. Zet de interrupt-vector ($EA31) drie bytes naar boven en de uitlezing van het toetsenbord wordt overgeslagen:

```
POKE 788, PEEK(788)+3  of  POKE 788,52
```

U komt weer in de standaard modus met

```
POKE 788, PEEK(788)-3  of  POKE 788,49
```

Als de functie van de STOP-toets is uitgeschakeld, kan hij als een gewone toets worden ingelezen in BASIC:

```
100 GET A$: IF A$=" " THEN 100
110 IF A$=CHR$(3) THEN PRINT "STOP-TOETS INGEDRUKT"
```

Zo keert een programma met de STOP-toets naar het menu terug.

Ook met RESTORE kan een programma worden onderbroken. Die toets schakelen we uit door de NMI-vector (wijst op $FE47) op $FEC1 (RTS) te zetten:

```
POKE 792,193  NMI-routine uit
```

U komt weer in de standaard modus met

```
POKE 792,71   NMI-routine aan
```

Met de STOP-vector (wijst op $F6ED) schakelen we RUN/STOP en RESTORE in een keer uit:

```
POKE 808,254
```

Bovendien resulteert deze POKE in een verminkte listing, maar dat heeft op het verloop van de routine geen effect. De graad van verminking hangt af van de lengte van het programma. In de standaard modus komt u weer met

```
POKE 808,237
```

Na deze POKEs kan een programma nog worden onderbroken door een hardware-reset van buitenaf. Een BASIC-programma kan dan niet

meer zomaar worden gelist. Ook deze reset kan met software worden uitgeschakeld (zie hoofdstuk 7).

Er zijn nog meer gevaarlijke toetsen. Door SHIFT en **C=** tegelijk in te drukken wisselt de computer van tekens. Met

POKE 657,128

verhindert u dat.

POKE 657,0

brengt u in de standaard modus. In de onderstaande tabel staan nog meer interessante POKE-instructies.

EFFECT	POKE-INSTRUCTIE
STOP uit	POKE 788, 52: POKE 808,239
STOP aan	POKE 788, 49: POKE 808,237
STOP, RESTORE, LIST uit	POKE 808,234
STOP, RESTORE, LIST uit	POKE 808,225
STOP, RESTORE, LIST aan	POKE 808,237
RESTORE uit	POKE 793,203
RESTORE uit	POKE 792,193
SAVE uit	POKE 818, 32: POKE 819,245
SAVE aan	POKE 818,237: POKE 819,245
LIST uit	POKE 775,200
LIST aan	POKE 775,167
TOETSENBORD uit	POKE 649, 0
TOETSENBORD aan	POKE 649, 10

3.9 Machinetaal ... hoewel?

Wie een beetje vakkennis heeft en met een machinetaalmonitor weet om te gaan, omzeilt elke LIST- of kopieerbeveiliging in BASIC. Velen wagen zich niet aan programma's in machinetaal. Omdat ze machinetaal niet begrijpen, denken ze dat je daar niets aan kunt veranderen. Bovendien stijgt de waarde van een programma vaak,

als men denkt dat het in machinetaal geschreven is. De routine in deze paragraaf simuleert daarom een spelletje in machinetaal, terwijl het eigenlijk om BASIC gaat. In zo'n programma zit maar een BASIC-regel. Daarin staat een spronginstructie naar een machinetaalprogramma, een SYS-instructie dus. Lees die eerst in. De tekst en regelnummers in het volgende voorbeeld mag u veranderen:

```
1985 SYS (2110) BY DATA BECKER
  10 SYS 2110
```

Zet daar de volgende BASIC-regels achter:

```
POKE 43,80: POKE2127,0:NEW
10 FOR I=2110 TO 2126
20 READ A
30 POKE I,A
40 NEXT
50 DATA 169, 80,133, 43,169, 52,141, 20,  3
60 DATA 169,193,141, 24,  3, 76,113,168
```

Omdat het er maar een paar zijn, worden de data niet gecontroleerd. Let er dus op, dat u de dataregels juist overneemt. Het programma in machinetaal:

```
#$80 : NIEUWE BASIC-START
0840 STA $2B   : INSTELLEN
0842 LDA #$34
0844 STA $0314 : RUN/STOP UIT
0847 LDA #$C1
0849 STA $0318 : RESTORE UIT
084C JMP $A871 : OPROEP RUN-ROUTINE
```

Start het programma met RUN en wis het na afloop. Daarna kan het programma dat moet worden beveiligd, geladen worden. Dat moet in regel 0 beginnen. Lees vervolgens

```
POKE 43,1
```

in en sla het spelletje gewoon op. Ook het laden verloopt normaal, maar bij LIST verschijnt alleen de regel met SYS. Als het programma met RUN is gestart, springt de computer naar de routine, die het geheugen verhoogt en RUN, STOP en RESTORE uitschakelt. Eenmaal gestart kan het aldus beveiligde BASIC-programma niet worden onderbroken. De beveiliging is natuurlijk niet 100% waterdicht, maar samen met de voorstellen uit andere paragrafen wordt het kraken van een programma toch wel erg moeilijk.

4 ZELF INSTRUCTIES UITBREIDEN

Op een aantal manieren kunnen instructies aan het bedrijfssysteem worden toegevoegd. In dit hoofdstuk behandelen we er drie:

1 - Veranderen van de BASIC-LINKS.
2 - CHRGET veranderen.
3 - IRQ-routine veranderen.

4.1 BASIC-LINKS veranderen

Dit is de meest effectieve manier om verschillende instructies in te voegen. Zet de vector ($0308-$0309 (776-777)), die wijst op de routine die BASIC-instructies interpreteert, op een machinetaalprogramma. We nemen aan, dat dat programma in de cassettebuffer ligt ($033C-$03FB):

```
033C LDA #$47
033E STA $0308
0341 LDA #$03
0343 STA $0309
0346 RTS
--------------
0347 JSR $0073
034A CMP #$5F
034C BEQ $0351
034E JMP $A7E7
0351 LDA #$05
0353 STA $0286
0356 JMP $A7E4
```

De routine bestaat uit twee delen:

DEEL 1 ($033C-$0346) zet BASIC-LINK op DEEL 2
DEEL 2 ($0347-$0356) is de instructie-uitbreiding:

JSR $0073: De computer haalt volgende teken van invoer ($0073=CHRGET-routine).

CMP #$5F: Dit teken wordt vergeleken met de ASCII-waarde voor "pijl naar links".

JMP $A7E7: Omdat de tekens niet gelijk zijn, springt de computer naar het adres, waar de vector gewoonlijk naar wijst. Daar wordt onderzocht of het ingevoerde teken een normale BASIC-instructie is.

LDA #$05:
STA $0286: De instructie-uitbreiding: de kleur van de tekens wordt groen.

JMP $A7E4: De computer keert terug in standaard modus.

Het machinetaalprogramma in BASIC:

```
10 FOR X= 828 TO 856: READ A: POKE X,A: NEXT X
20 DATA 169, 71,141,  8,  3,169,  3,141,  9,  3, 96
30 DATA  32,115,  0,201, 95,240,  3, 76,231,167,169
40 DATA   5,141,134,  2, 76,228,167
RUN

SYS828
```

De tekens worden groen, als u op <- heeft gedrukt en RETURN heeft gegeven. Bij verschillende uitgebreide instructies achter elkaar heeft de computer heel veel tijd nodig om ze na te lopen. Dat kan worden vermeden door aan alle nieuwe BASIC-instructies dezelfde markering vooraf te laten gaan, bijvoorbeeld <- of !. Als in een programma voor instructie-uitbreiding de markering niet wordt gevonden, springt de computer naar de normale decodering van het toetsenbord. Anders gaat hij naar een nieuwe instructie. Een programmavoorbeeld ter verduidelijking:

```
033C  LDA #$47
033E  STA $0308
0341  LDA #$03
0343  STA $0309
0346  RTS
0347  JSR $0073
034A  CMP #$21    : ASCII-WAARDE VOOR "!"
034C  BEQ $0351   : NIEUWE INSTRUCTIE AANWEZIG
034E  JMP $A7E7   : GEEN NIEUWE INSTRUCTIE
0351  JSR $0073   : LEEST TEKEN NA !
0354  CMP #$91    : ON?($91=TOKEN VOOR ON)
0356  BEQ $035F   : JA
0358  CMP #$A8    : NOT?($A9=TOKEN VOOR NOT)
035A  BEQ $0374   : JA
035C  JMP $AF08   : ONJUISTE INSTRUCTIE, SYNTAX ERROR
035F  SEI         : ON-INSTRUCTIE
0360  LDA #$6E    : ZET INTERRUPT OP $036E
```

```
0362  STA $0314
0365  LDA #$03
0367  STA $0315
036A  CLI
036B  JMP $A7E7   : ON-INSTRUCTIE AFSLUITEN
036E  INC $0286   : NIEUWE IRQ
0371  JMP $EA31   : OUDE IRQ VERDER
0374  SEI         : NOT-INSTRUCTIE
0375  LDA #$31    : IRQ WEER NORMAAL
0377  STA $0314
037A  LDA $EA
037D  STA $0315
0380  CLI
0381  JMP $A7E7   : NOT-INSTRUCTIE AFSLUITEN
```

De routine in BASIC:

```
10 FOR X= 828 TO 898: READ A: POKEX,A: NEXT X
20 DATA 169, 71,141,  8,  3,169,  3,141,  9,  3, 96
30 DATA  32,115,  0,201, 33,240,  3, 76,231,167, 32
35 DATA 115,  0,201,145,240,  7,201,168,240, 24, 76
40 DATA   8,175,120,169,110,141, 20,  3,169,  3,141
45 DATA  21,  3, 88, 76,228,167,238,134,  2, 76, 49
60 DATA 234,120,169, 49,141, 20,  3,169,234,141, 21
65 DATA   3, 88, 76,228,167
```

Na RUN en SYS 828 staan er twee nieuwe instructies ter beschikking: !ON zet de IRQ-vector op $036E. Elke 1/60 seconde wordt de kleur van de tekens verhoogd. !NOT stopt dit. U ziet, dat oude instructies nieuwe functies kunnen krijgen zonder dat de functie van de oorspronkelijke instructies verandert.

U kunt het beste de instructies in een tabel zetten en ze vergelijken met de ingevoerde instructies. Zo werkt de BASIC-interpreter ook. Door van nieuwe instructies tokens te maken bespaart u geheugenruimte. De betreffende vector ligt op $0304-$0305 (772-773). Nog vrij zijn de tokens 202-254.

4.2 CHRGET-routine veranderen

Dit is de CHRGET-routine ($0073-$0084):

```
0073 INC $007A
```

```
0075 BNE $0079
0077 INC $007B
0079 LDA $HHLL    : HAALT TEKEN UIT BASIC-TEKST
007C CMP #$3A
007E BCS $008A
0080 CMP #$20
0082 BEQ $0073
0084 SEC
0085 SBC #$30
0087 SEC
0088 SBC #$D0
008A RTS
```

Omdat de routine bij het inschakelen van de computer van het ROM naar het RAM wordt gekopieerd, kan ze worden gewijzigd. Vector $0073-$007B kan zo blijven, in $007C moet JMP $(eigen programma) staan. In dit geval controleert dat programma of het teken dat wordt gelezen, een nieuwe instructie is. Zo niet, dan moet de CHRGET-routine worden uitgevoerd. Als voorbeeld geven we NEW de betekenis RESET. Deze routine ligt in de cassettebuffer:

```
033C LDA #$4C
033E STA $007C
0340 LDA #$49
0342 STA $007D
0344 LDA #$03
0346 STA $0076
0348 RTS
0349 CMP #$A2  : NEW?
034B BEQ #035F : JA
034D CMP #$3A  : NEE, MAAR CHRGET CONTINUEREN
034F BCS $035E
0351 CMP #$20
0353 BNE $0358
0355 JMP $0073
0358 SEC
0359 SBC #$30
035B SEC
035C SBC #$D0
035E RTS
035F JMP $FCE2
```

Het eerste deel verandert de CHRGET-routine. Bij elke sprong naar die routine wordt het tweede deel doorlopen. De BASIC-lader:

```
10 FOR X=828 TO 865: READ A: POKE X,A: NEXT X
20 DATA 169, 76,133,124,169, 73,133,125,169,  3,133,126, 96
30 DATA 201,162,240, 18,201, 58,176, 13,201, 32,208,  3, 76
40 DATA 114,  0, 56,233, 48, 56,233,208, 96, 76,225,252
```

Als u vervolgens NEW intoetst, voert de computer een RESET uit.

4.3 IRQ-vector veranderen

Deze mogelijkheid wordt niet vaak gebruikt. Dat er elke 1/60 seconde naar de interruptroutine (IRQ) wordt gesprongen, kunnen we uitbuiten:

```
033C SEI
033D LDA #$49
033F STA $0314
0342 LDA #$03
0344 STA $0315
0347 CLI
0348 RTS
0349 LDA $CB
034B CMP #$39
034D BEQ $0352
034F JMP $EA31
0352 JSR $E544
0355 JMP $EA31
```

In BASIC:

```
10 FOR X= 828 TO 855: READ A: POKE X,A: NEXT X
20 DATA 120,169, 73,141, 20,  3,169,  3,141, 21,  3, 88
30 DATA  96,165,203,201, 57,240,  3, 76, 49,234, 32, 68
40 DATA 229, 76, 49,234
```

Na RUN en SYS 828 gebeurt er niets eigenaardigs, maar met <- wist u het beeldscherm. Soms verdwijnt de cursor, als u die toets gebruikt. Dat is maar tijdelijk: bij het schrijven wordt hij weer zichtbaar.

De IRQ-vector wordt op de eigen routine gezet, die controleert of een bepaalde toets is ingedrukt (hier: <-). Zo ja, dan wordt het scherm gewist. In elk geval moet u terug naar de IRQ-routine. Een

groot verschil is, dat de mogelijkheden in paragraaf 4.1 en 4.2 de instructie pas accepteren na RETURN. Bij de variant in deze paragraaf reageert de computer direkt.

5 GRAFIEK

5.1 Basiskennis

De vorm van alle tekens die via het toetsenbord op het scherm verschijnen, moet gedefinieerd zijn. Het daarvoor gereserveerde geheugenbereik ($D000-$DFFF in het ROM) heet karakter- of tekengenerator. Elk teken kan worden weergegeven in een matrix van 8*8, bestaat dus uit maximaal 64 punten. De A ziet er zo uit:

```
     76543210

1    ...**...
2    ..****..
3    .**..**.
4    .******.
5    .**..**.
6    .**..**.
7    .**..**.
8    ........
```

Elke regel is een byte en elke plaats een bit. Daarmee is de vorm van een teken gedigitaliseerd. Voor de berekening van de waarde van een byte telt u de bits op van de punten, die gezet zijn. De bytewaarde van de derde regel van de A is dus 102:

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	grondtal^{macht}
128	64	32	16	8	4	2	1	
.	*	*	.	.	*	*	.	gezette punten
0+	64+	32+	0+	0+	4+	2+	0	= 102

Ook het volgende programma laat duidelijk zien hoe de digitalisering in zijn werk gaat:

De gebruikte variabelen zijn:

J - waarde van actuele bit: (2)
EG$ - berekende regel
BY - bytewaarde

```
10 REM TRAININGSPROGRAMMA KARAKTERGENERA
TOR
20 PRINTCHR$(147):POKE53280,1:POKE53281,
1:POKE646,6:REM KLEUR
30 CLR:PRINT"▣ BYTE - BIT - TRAININGSPRO
GRAMMA          ■":PRINT
40 PRINT"[1] REGEL INVOEREN":PRINT
50 PRINT"[2] BYTE INVOEREN":PRINT:PRINT
60 PRINT"TYPE 1 OF 2 !"
70 GET A$:IF A$="" THEN 70
80 IF A$="1" THEN 200
90 REM *** BYTE INVOEREN ***
100 PRINTCHR$(147)
110 INPUT"DE BYTE-WAARDE (0-255)";BY:PRI
NT:IF BY<0 OR BY>255 THEN 110
120 FOR A=7 TO 0 STEP-1
130 J=2↑A
140 IF J<=BY THEN PRINT"●";:BY=BY-J:GOTO
160
150 PRINT"◣O▀";
160 NEXT
170 PRINT"▩▩▩DRUK EEN TOETS !"
180 GETA$:IFA$=""THEN180
190 GOTO20
200 REM *** REGEL INVOEREN ***
210 PRINTCHR$(147)
220 PRINT"X=PUNT GEZET   M=PUNT NIET GEZ
ET":PRINT:PRINT
230 FOR A=7 TO 0 STEP-1
240 PRINT8-A"PUNT ?";
241 GET A$:IF A$<>"M" THEN IF A$<>"X" TH
EN 241
250 PRINT A$:IF A$="X" THEN BY=BY+2↑A:EG
$=EG$+"●":GOTO 270
260 EG$=EG$+"◣O▀"
270 NEXT A
280 PRINT:PRINTEG$"="BY
290 PRINT:PRINT"DRUK EEN TOETS !"
300 GETA$:IFA$=""THEN300
310 GOTO20
```

Na het starten van het programma kan een regel of een byte worden ingevoerd. Kiest u de regel, dan vraagt de computer naar de 8 plaatsen. Met een X zet u een punt, elke andere letter is een lege ruimte. Zijn de plaatsen gedefinieerd, dan verschijnt de regel op het scherm en wordt de bytewaarde berekend. Als u een bytewaarde inleest (0-255), levert de computer de regel.

5.2 De karaktergenerator in het geheugen

Het volgende diagram geeft de plaats aan van de karaktergenerator in het ROM:

KARAKTERSET 1

$D000-$D400 (53248-54272) hoofdletters/grafiek: normaal

$D400-$D800 (54272-55296) hoofdletters/grafiek: reverse

KARAKTERSET 2

$D800-$DC00 (55296-56320) hoofd- en kleine letters: normaal

$DC00-$DFFF (56320-57343) hoofd- en kleine letters: reverse

Elk teken neemt 8 bytes in beslag (1 byte per regel). Een karakterset bestaat uit 256 tekens: 128 gewone en 128 reverse. In totaal zijn er dus 512 verschillende tekens. Ze nemen in het geheugen 512*8=4 Kbyte in beslag. Met

```
10 FOR A=0 TO 255: POKE 1024+A,A: POKE 55296+A,1: NEXT
```

verschijnt de tweede set op het scherm.

5.3 Karakterset uitlezen

Om de karaktergenerator te kunnen veranderen moet hij van het ROM in het RAM worden gekopieerd. Dat kan gebeuren met de volgende lus:

```
10 DIM  B(4095)
20 FOR A=0 TO 4095: REM $D000- $DFFF
30 B(A)=PEEK(53248+A)
40 NEXT A: REM INLEZEN BEEINDIGEN
```

```
50 FOR A= 0 TO 4095
60 PRINT B(A);: NEXT A
70 END
```

Het uitlezen van het geheugen duurt ongeveer een minuut. Daarna wordt er een flinke hoeveelheid getallen uitgevoerd, voor het grootste deel nullen en getallen boven de 240. Dat is niet de inhoud van de karaktergenerator maar een onderdeel van het RAM, die beide $D000-$DFFF gebruiken. In dat RAM-gedeelte zijn de VIC's, die bijvoorbeeld de sprites sturen, de SID (sound interface device) en het kleurgeheugen ondergebracht. Om bij het karakter-ROM te kunnen komen moet bit 2 van de processorpoort (adres 1) gewist en de interrupt (SEI) verhinderd worden. De computer kan dan niet meer bij het RAM-gedeelte. Voeg de volgende regels aan het vorige programma toe:

```
 5 POKE 56334,0: POKE 1,51: REM int. aan, kar.-ROM uit
45 POKE 1,55: POKE 56334,1: REM int. aan, kar.-ROM aan
```

Adres 56334 verhindert de interrupt. Loopt het programma eenmaal, dan kan het niet worden afgebroken door RUN/STOP en RESTORE tegelijk in te drukken. Pas op: als u POKE 1,51 inleest zonder de interrupt te hebben verhinderd, loopt de computer vast. Ongeveer een minuut nadat het programma is gestart, verschijnt de inhoud van het karakter-ROM. Voeg de volgende regels toe aan het oude programma (regels 50-70 worden vernieuwd) en de getallen veranderen in leesbare tekens.

Variabelen:

B(0)-B(4095)	-	inhoud van karaktergenerator
A	-	flag: 0=punt gezet
X	-	lus, haalt plaats van regel
2^X	-	bitwaarde
Z	-	karakterwaarde

```
50 REM * * * GETALLEN OMZETTEN IN LETTERS * * *
55 Z=B(I)
60 FOR X=7 TO 0 STEP -1
70 A=Z AND 2^X
80 IF A THEN PRINT ".";: GOTO 100: REM A=0, PUNT
90 PRINT" ";: REM GEEN PUNT
100 NEXT X: PRINT
110 I=I+1: IF I=4096 THEN END
120 GOTO 50
```

Na ongeveer een minuut is het geheugen uitgelezen; op het scherm staan vergrote tekens in plaats van getallen.

5.4 Karaktergenerator kopiëren

Nu bekend is hoe de karaktergenerator moet worden uitgelezen, kunt u hem zonder problemen van het ROM in het RAM kopiëren om veranderingen aan te brengen. Voor de exacte plaats zijn er verschillende mogelijkheden.

In het BASIC-geheugen

De kopie van de karaktergenerator kan aan het begin van het BASIC-geheugen worden gezet. De BASIC-start moet dan 2 Kbyte verschoven worden. Als u hem aan het einde van het BASIC-geheugen zet, hoeft dat niet te gebeuren.

Kopie bij BASIC-start:

```
POKE 44,16: POKE 16+256,0: NEW
```

```
10 POKE 56334,0: POKE 1,51:REM TOEGANG TOT CHAR-ROM
20 FOR K=0 TO 2047
30 POKE 2048+K, PEEK(53248+K)
40 NEXT
50 POKE 1,55: POKE 56334,1
60 POKE 53272,PEEK(53272) AND NOT 12 OR 2: REM OMSCHAKELEN
```

Begin karaktergenerator bij 2048 ($0800),
beeldscchermgeheugen bij 1024 ($0400)

Om interne redenen kan de karaktergenerator aan het begin van BASIC maar over 2 Kbyte beschikken. Er is dus plaats voor 1 van de 2 karaktersets, in dit geval nummer 1. Voor nummer 2 dient u regel 30 te veranderen in:

```
30 POKE 2048+K, PEEK(55296+K)
```

Onder het ROM

Is het bedrijfssysteem niet in het RAM gezet, dan zijn er onder het ROM 8 Kbyte vrij. Omdat de PEEK-instructie het ROM aanspreekt, kan een karakterset, die in het RAM onder het ROM is gekopieerd, alleen worden uitgelezen met de gemodificeerde PEEK-instructie uit TIPS & TRUCS 1. Dat dat met de originele PEEK niet kan, maakt in feite natuurlijk weinig uit.

Bij de karaktergenerator onder het ROM is de BASIC-start is niet relevant.

```
10 POKE 56334,0:POKE 1,51:REM TOEGANG TOT CHAR-ROM
20 FOR K=0 TO 4095
30 POKE 57344+K, PEEK(53248+K)
40 NEXT K
50 POKE 1,55: POKE 56334,1
60 POKE 56576,196: REM HOOGSTE 16-K-BLOK
70 POKE 648,3*64+4: REM SCHERM-RAM NAAR (3*64+4)*256
80 POKE 53272,PEEK(53272) AND NOT 6 OR 8: REM OMSCHAKELEN
```

In de volgende voorbeelden is de karaktergenerator gekopieerd met het tweede programma. Let op de volgende zaken:

- Het beginadres van het nieuwe beeldschermgeheugen is 50176 en niet 1024.

- POKE 56576,199: POKE 648,4: POKE 53272,21 brengt u weer terug in de standaardmodus

- Door het intoetsen van RUN/STOP & RESTORE komt u alleen volledig in de standaardmodus terug met een extra POKE: POKE 648,4.

Lees het tweede kopieerprogramma in en start het. Na circa 60 seconden verschijnen cursor en karakterset. Het eerste teken in de karaktergenerator is de apestaart (57344-57351). Door

```
POKE 57344,255
```

wordt dit teken platter aan de bovenkant. Met

```
POKE 57351,255
```

gebeurt dat ook aan de onderkant.

5.5 Karaktergenerator omschakelen

Voor het maken van de kopie zijn de adressen 53272, 56576 en 648 nodig. Adres 53272 geeft in eerste instantie de geheugenplaats aan van de nieuwe karakterset. De bits 1-3 regelen de verschuiving van de karakterset. Bit 0 wordt niet gebruikt, de bits 4-7 verschuiven het beeldschermgeheugen.

De mogelijke beginadressen en hun bitcombinaties:

beeldschermgeheugen			karakterset		
0000xxxx	0		xxxx000x	0	
0001xxxx	1024	(normaal)	xxxx001x	2048	
0010xxxx	2048		xxxx010x	4096	
0011xxxx	3072		xxxx011x	6144	(normaal)
0100xxxx	4096		xxxx100x	8192	
0101xxxx	5120		xxxx101x	10240	
0110xxxx	6144		xxxx110x	12288	
0111xxxx	7168		xxxx111x	14336	
1000xxxx	8192				
1001xxxx	9216				
1010xxxx	10240				
1011xxxx	11264				
1100xxxx	12288				
1101xxxx	13312				
1110xxxx	14336				
1111xxxx	15360				

Als u het beeldschermgeheugen verschuift, moet ook het MSB van de video-RAM worden veranderd:

```
POKE 648,(beginadres)*256
```

Uit het overzicht valt op te maken, dat karakterset en beeldschermgeheugen alleen binnen de eerste 16 Kbyte kunnen worden verschoven. De vier adresbits van adres 56576 maken een verschuiving in het hele geheugen mogelijk. Wordt een ander gebied van 16K gekozen dan het eerste, dan verschuift het beeldschermgeheugen mee, maar ook de spriteblocks en hun wijzers.

16K-bereik

```
$0000-$3FFF     0-16383 POKE 56576,199: POKE 648,4
$4000-$7FFF 16384-32767 POKE 56576,198: POKE 648,4+1*64
$8000-$BFFF 32768-49151 POKE 56576,197: POKE 648,4+2*64
$C000-$FFFF 49152-65535 POKE 56576,196: POKE 648,4+3+64
```

Omdat het ontwerpen van een hele karakterset met POKEs een martelgang is, staan er op de volgende bladzijden twee listings, die dat proces sterk vereenvoudigen.

5.6 Hulpprogrammatuur voor de definitie van karaktersets

De volgende karaktereditor biedt zeer veel mogelijkheden. Zo kan een teken 90, 180 of 270 graden worden gedraaid, geïnverteerd, verwisseld, gedupliceerd enzovoort. Voorkennis van de karaktergenerator en aanverwante zaken is niet nodig. De tekens worden gemaakt met het toetsenbord of joystick 2. Karaktersets kunnen op disk worden opgeslagen. Aan het eind van het programma staat een ingebouwde programmagenerator, die de BASIC-lader voor de nieuwe karakterset (ten hoogste 256 tekens) kan schrijven.

```
5 REM PROGRAM BY T.WELTNER*BAHNHOFSALLEE
 10*3260 RINTELN 1*TEL.:(05751) 5676
10 POKE45,255:POKE46,66:CLR
20 POKE788,52:O=53248:POKEO+32,1:POKEO+3
3,1:PRINT"■■":POKE214,4:PRINT
30 IFPEEK(51000)=1THENPOKE51000,0:GOTO35
0
40 POKE53247,0:PRINTTAB(7)"■
         ":PRINT
50 POKEO+42,3:PRINTTAB(7)"■ * TEKEN-EDIT
OR *":PRINT
60 PRINTTAB(7)"■■                      ":
POKE214,16:PRINT
70 PRINT"■■■■[ VERANDEREN VAN KARAKTERSE
T":PRINT
80 PRINT"■■■■[ INGEBOUWDE PROGRAMMAGENER
ATOR"
90 POKE214,21:PRINT:PRINT"■■■■■BY T.WELT
NER / 22.SEPTEMBER 1984 "
110 :
120 REM *** SPRITE INITIALISEREN ***
130 FORK=0TO7:POKE704+K*3,255:POKE705+K*
3,0:POKE706+K*3,0:NEXTK
140 FORK=0TO38:POKE704+8*3+K,0:NEXTK
150 FORK=0TO63:POKE832+K,0:NEXT:REM BLOC
K 13 FULL
170 :
180 REM *** CURSOR FLASH ***
190 READA:IFA=-1THEN240
200 POKE912+B,A:C=C+A:B=B+1:GOTO190
210 DATA174,142,3,232,224,16,240,6,142,1
42,3,76,52,234,169,0,141,142
220 DATA3,173,143,3,201,4,240,11,141,21,
208,169,4,141,143,3,76,52,234
```

```
230 DATA141,21,208,169,0,76,175,3,120,16
9,144,160,3,141,20,3,140,21,3,88,96,-1
240 IFC<>6009THENLIST200-230
260 :
270 C=0:REM *** KOPIEER-ROUTINE ***
280 FORA=0TO48:READB:POKE12*4096+A,B:C=C
+B:NEXT
290 DATA162,16,169,0,141,14,220,169,51,1
33,1,169,208,160,0,132,34,133
300 DATA35,169,112,132,36,133,37,177,34,
145,36,200,208,249,230,35,230
310 DATA37,202,208,242,169,55,133,1,169,
1,141,14,220,96
320 IFC<>5798THENPRINT"? DATA FOUT":LIST
290-310
330 SYS49152:REM ACTIVEREN
350 :
360 CLR:O=53248:POKE198,0:REM *** MENU *
**
370 PRINT"                    * MENU *
          "
380 PRINT"  1    EEN TEKEN PRODUCEREN"
:PRINT
400 PRINT"  2    GEGEVENS INVOEREN":PR
INT
410 PRINT"  3    OVERZICHT TEKENS":PRI
NT
420 PRINT"  4    TEKENS WISSELEN":PRIN
T
430 PRINT"   ";:FORT=0TO34:PRINT"=";:NEX
TT:PRINT:PRINT
440 PRINT"  5    KARAKTERSET LADEN":PR
INT
450 PRINT"  6    KARAKTERSET OPSLAAN":
PRINT
455 PRINT"  7    PROGRAMMA-GENERATOR":
PRINT
456 PRINT"   ";:FORT=0TO34:PRINT"=";:NEX
TT:PRINT:PRINT
460 PRINT"  8    PROGRAMMA BEEINDIGEN"
470 GETA$:IFA$=""THEN470
480 IFA$=CHR$(3)THENPRINT"? BREAK IN ME
NU";:END
490 A=VAL(A$):IFA=0ORA>8THENGOTO470
500 ONAGOTO530,590,800,1020,1490,1270,33
90,1230
520 :
530 REM *** TEKEN-EDITOR ***
```

```
540 GOSUB1790: REM TEKEN INLEZEN
550 GOSUB1970:GOTO2150:REM SCHERM INITIA
LISEREN/BEWEGINGSROUTINE
580 :
590 REM *** GEGEVENS INVOEREN ***
600 PRINT"                 GEGEVENS INVOER
EN          "
610 PRINT:PRINT:PRINT"GEEF DE 8 GEGEVEN
S IN!"
620 PRINT:PRINT:FORVV=0TO7:PRINTVV+1"";
:K(VV)=0:K$(VV)="":INPUT".";K$(VV):NEXT
621 FORVV=0TO7:K(VV)=VAL(K$(VV)):IFK(VV)
<0ORK(VV)>255THENK(VV)=0
622 NEXTVV
630 F4=5:F3=1:FL=12:GOSUB3210:F3=0:FL=0:
F4=0
640 POKE214,15:PRINT:PRINT"MOET DIT TEKE
N OPGESLAGEN WORDEN (J/N) ?"
650 GETA$:IFA$=""THEN650
660 IFA$="J"THENKL$="TEKENS INVOEREN":G
OTO720
670 PRINT:PRINT"TERUG NAAR MENU (J/N)?"
680 GETA$:IFA$=""THEN680
690 IFA$="J"THENGOTO360
700 GOTO590
705 :
710 REM *** INVOER ***
720 P=53246:PRINT"";KL$:PRINT"TEKEN
?";:GOSUB785
721 POKE204,0:GETA$:IFA$=""THEN721
722 A=PEEK(207):IFATHEN722
723 PRINT:IFA$=CHR$(13)THENA$=" "
724 POKE204,1:PRINT"TE VERANDEREN TEKE
N: ";A$:FORT=0TO24:PRINTCHR$(101);
725 NEXTT:PRINT:PRINT:PRINT" 1 ";E
$
726 PRINT:PRINT" 2 ";F$
727 PRINT:PRINT" 3 AUTO ";X$
728 PRINT:PRINT:PRINT" 4 FOUT"
729 PRINT:PRINT:PRINT" 5 O.K., VER
DER"
731 GETC$:IFC$=""THEN731
733 IFC$="5"THEN750
734 IFC$="3"THENGOSUB780
735 IFPEEK(P-1)=1THENGOTO739
```

```
737 IFC$="1"THENRV=RV+128:POKE214,5:PRIN
T:PRINTTAB(5)"NEGA":POKEP-1,1:GOTO731
739 IFC$="1"THENRV=RV-128:POKE214,5:PRIN
T:PRINTTAB(5)"POSI":POKEP-1,0:GOTO731
741 IFPEEK(P-2)=1THENGOTO745
743 IFC$="2"THENRV=RV+256:POKE214,7:PRIN
T:PRINTTAB(5)"KLEIN/GROOT":N=1:GOTO790
745 IFC$="2"THENRV=RV-256:POKE214,7:PRIN
T:PRINTTAB(5)"GROOT/GRAF.":N=0:GOTO790
747 IFC$="4"THENGOTO720
750 POKE646,1:PRINT"██"A$:POKE646,0:CO=P
EEK(1024+40)+RV:IFCH<>1THENGOSUB3380
760 IFCH=1THENRETURN
770 POKE214,20:PRINT:PRINT"OK   ":PRINT"
      ]":GOTO670
780 IFPEEK(P)=0THENGOTO782
781 AU$="":POKE214,9:PRINT:PRINTTAB(5)"█
████UIT":POKEP,0:GOTO731
782 AU$="A":POKE214,9:PRINT:PRINTTAB(5)"
█████AAN":POKEP,1:GOTO731
785 RV=0:X$="UIT":E$="POSITIEF":F$="GROO
T/GRAF.":IFPEEK(P)=1THENX$="AAN"
786 IFPEEK(P-1)=1THENE$="NEGATIEF":RV=RV
+128
787 IFPEEK(P-2)=1THENF$="KLEIN/GROOT":RV
=RV+256
788 RETURN
790 POKEP-2,N:GOTO731
791 :
800 MD=189:REM *** OVERZICHT ***
810 POKE53272,MD:POKE56576,150:POKE648,1
08:PRINT"]":AQ=55296:FORA=0TO255
820 POKE27648+2*A,A:POKEAQ+2*A,14:IFCH<>
1THENPOKE27648+A+600,A:POKEAQ+A+600,3
829 NEXTA:IFCH=1THENRETURN
830 GETA$:IFA$=""THEN830
840 POKE53272,21:POKE56576,151:POKE648,4
841 GOTO360
1000 :
1020 REM *** WISSELEN ***
1030 PRINT"]██                TEKENS WISSELE
N              "
1040 PRINT"███ 1 ████TEKENS KOPIEREN"
1041 PRINT"███ 2 ████TEKEN WISSELEN"
1042 PRINT"████ 3 ████MENU"
1050 GETQ$:IFQ$=""THEN1050
1055 IFVAL(Q$)<1ORVAL(Q$)>2THENPOKE51000
,1:RUN
```

```
1060 IFQ$="2"THENKL$="▒WISSELEN: A <-> B
":GOTO1120
1100 KL$="▒KOPIEREN: A(A) -> A(B)"
1120 CH=1:GOSUB720:CQ=CO:GOSUB720:CH=0:P
RINT
1130 IFQ$="2"THENGOSUB1190:GOTO1150
1140 FORV=0TO7:POKE28672+CO*8+V,PEEK(286
72+CQ*8+V):NEXTV
1150 POKE214,15:PRINT:PRINT"TERUG NAAR M
ENU (J/N) ?"
1160 GETA$:IFA$=""THEN1160
1170 IFA$="J"THENGOTO360
1180 PRINT"▯":GOTO1060
1190 FORV=0TO7:C1(V)=PEEK(28672+CQ*8+V):
C2(V)=PEEK(28672+CO*8+V):NEXTV
1200 FORV=0TO7:POKE28672+CQ*8+V,C2(V):PO
KE28672+CO*8+V,C1(V):NEXTV:RETURN
1210 :
1230 REM *** PROGRAMMA BEEINDIGEN ***
1240 PRINT"▯TOT DE VOLGENDE KEER...":POK
E788,49:PRINT"▒▒▒NEW▯▯▯":END
1250 :
1270 REM *** OPSLAAN ***
1280 PRINT"▯▒▯        KARAKTERSET OPSLAAN
                "

1290 PRINT"▒▒▒DE DOOR U VERANDERDE KARAK
TERSET VANAF"
1300 PRINT"ADRES (DEC.) 28672 KAN NU OPG
ESLAGEN"
1310 PRINT"WORDEN.":PRINT:IFHI=0THENHI=1
12:HII=128
1320 PRINT"HIJ BESLAAT OP DISKETTE CA. 1
7(9) KBYTE."
1330 PRINT"OM PROGRAMMATECHNISCHE REDENE
N"
1340 PRINT"KAN DE KARAKTERSET NIET OP DA
TASETTE"
1350 PRINT"OPGESLAGEN WORDEN.":PRINT:PRI
NT
1360 IFPEEK(186)<>8THENPRINT"▒ALLEEN VOO
R DISKETTE!":FORT=1TO999:NEXTT:GOTO360
1380 PRINT:PRINT"ONDER WELKE NAAM MOET D
E KARAKTERSET"
1385 PRINT"WORDEN OPGESLAGEN?"
1390 A$="":INPUT A$:IFA$=""THENGOTO 360
1395 IFA$="PARAMETER"ORRIGHT$(A$,3)="PAR
AMETER"THENGOTO1470
1400 IFKE=0THENIFLEFT$(A$,1)<>CHR$(215)T
HENB$=CHR$(215)+A$:GOTO1420
```

```
1410 B$=A$
1420 PRINT""
1430 PRINT"POKE43,0:POKE44,";H2;":POKE45
,0:POKE46,";H4;""
1440 PRINT"SAVE"CHR$(34)"@:"B$CHR$(34)",
8"
1450 PRINT"POKE43,1:POKE44,8:POKE51000,1
:GOTO10":PRINT""
1460 FORA=0TO7:POKE631+A,13:NEXT:POKE198
,8:END
1470 POKE646,12:PRINT"
PARAMETER                 "
1471 PRINTW1$:PRINT:PRINT" 1    1/1 SAV
EN":PRINT:PRINT" 2    1.HELFT":PRINT
1472 PRINT" 3    2.HELFT":PRINT:PRINT"
 4    IDENTIFICATIE UIT"
1473 PRINT:PRINT" 5    FOUT":PRINT:PRIN
T" 6    O.K., VERDER"
1474 GETW$:IFW$=""THEN1474
1475 IFW$="5"THENKE=0:H2=112:H4=128:POKE
53247,0:W2$="":W3$="":GOTO1470
1476 IFW$="2"THENH2=112:H4=120:W2$="1.HE
LFT OPSLAAN"
1477 IFW$="3"THENH2=120:H4=128:W2$="2.HE
LFT OPSLAAN"
1478 IFW$="4"THENPOKE53247,1:W3$="IDENTI
FICATIE"
1479 IFW$="1"THENH2=112:H4=128:W2$="1/1
SAVEN........"
1480 IFW$="6"THENGOTO1280
1481 PRINT"";W2$;"";W3$
1482 W1$="":GOTO1471
1483 :
1490 REM *** KARAKTERSET LADEN ***
1500 PRINT"            KARAKTERSET LADEN
            "

1520 PRINT"U KUNT NU EEN REEDS EERD
ER OPGESLAGEN";
1530 PRINT"KARAKTERSET VAN DISK LADEN":P
RINT:PRINT"('$' = DIRECTORY)"
1540 A$="":PRINT:PRINT:PRINT:INPUT"NAAM
VAN KARAKTERSET";A$
1550 IFA$=""THENGOTO360
1560 IFA$="$"ANDPEEK(186)=8THENGOSUB1640
:PRINT:GOTO1540
1570 IFPEEK(53247)=0THENIFLEFT$(A$,1)<>C
HR$(215)THENB$=CHR$(215)+A$:GOTO1600
1590 B$=A$
1600 IFPEEK(186)=1THENPRINT"ALLEN VOOR
```

```
DISKGEBRUIK":FORT=0TO999:NEXTT:GOTO360
1610 FORA=0TO8:POKE631+A,13:NEXT:PRINT"
"
1620 PRINT"":PRINT"LOAD"CHR$(34)B$CH
R$(34)",8,1"
1630 PRINT"POKE51000,1:GOTO10":PRINT"":
POKE198,8:END
1640 PRINT" O=KARAKTERSET"
1650 PRINT:OPEN1,8,15,"I0":OPEN2,8,2,"#"
:Q=18:W=1
1660 PRINT#1,"B-R";2;0;Q;W:PRINT#1,"B-P"
;2;0
1670 GET#2,X$:IFX$=""THENX$=CHR$(0)
1680 Q=ASC(X$)
1690 GET#2,X$:IFX$=""THENX$=CHR$(0)
1700 W=ASC(X$)
1710 FORX=0TO7:PRINT#1,"B-P";2;X*32+5:Y$
="":FORY=0TO15
1720 GET#2,X$:IFX$=""THENX$=CHR$(0)
1730 IFASC(X$)=160THEN1750
1740 Y$=Y$+X$:NEXTY
1750 PRINTY$:NEXTX:IFQ<>0THEN1660
1760 CLOSE1:CLOSE2:RETURN
1780 :
1790 REM *** TEKEN VERANDEREN ***
1810 KL$="               TEKEN-EDITOR
          "
1840 CH=1:GOSUB720:CH=0
1850 RETURN
1950 :
1960 REM *** BEELDSCHERMINITIALISERING *
**
1970 PRINT"                  TEKEN-EDITOR
           "
1980 RV$="OFF":P=53246:IFPEEK(P-1)THENRV
$="ON"
1981 ZS$="GROOT/GRAFIEK":IFPEEK(P-2)=1TH
ENZS$="KLEIN/GROOT"
1982 PRINT"TEKEN: ";A$;"REVERSE";RV$;"
";ZS$;"";:IFPEEK(P)=1THENPRINT"A";
2010 PRINT"76543210765432
10"
2020 PRINT"┌┬┬┬┬┬┬┬┐┌┬┬┬┬┬┬┬┐
"
2030 FORK=0TO7:PRINT""K"H"K"H
H"K:NEXTK
2040 PRINT"└┴┴┴┴┴┴┴┘└┴┴┴┴┴┴┴┘"
2050 PRINT"7654321076543210"
2060 PRINT" USER MATRIX
```

```
■■"
2070 IFFL=7THENRETURN
2080 PRINT"■■ F1  ■■ F3  ■■ F5  ■■ F7
■■■  F2  "
2100 PRINT"■■ TEST ■■ DATA ■■ ROM  ■■ RAM
■■ROTATE"
2140 :
2150 REM *** BEWEGINGSROUTINE ***
2160 POKE959,144:POKE961,3:SYS957:POKEO+
21,5:POKE2042,11
2170 SH=73:SV=106:BV=0:BH=0:REM SPRITE-P
OSITIE
2180 GETA$:POKE198,0:JO=PEEK(56320):IFA$
=""THENAS=0:GOTO2190
2185 AS=ASC(A$)
2190 IFAS=145OR(JOAND1)=0THENPOKE911,4:G
OSUB2450:REM CRSR UP
2200 IFAS=17OR(JOAND2)=0THENPOKE911,4:GO
SUB2480:REM CRSR DOWN
2210 IFAS=29OR(JOAND8)=0THENPOKE911,4:GO
SUB2510:REM CRSR RIGHT
2220 IFAS=157OR(JOAND4)=0THENPOKE911,4:G
OSUB2540:REM CRSR LEFT
2230 IFAS=19THENBH=0:BV=0:SH=73:SV=106
2240 IFAS=3THENPOKE959,52:POKE961,234:SY
S957:POKEO+21,0:GOTO360
2250 IFAS=133THENGOSUB2820: REM F1
2260 IFAS=134THENGOSUB2570: REM F3
2270 IFAS=137THENGOSUB3700: REM F2
2280 IFAS=135THENGOSUB2780: REM F5
2290 IFAS=139THENF2=12:FL=1:GOSUB3210:F2
=0:FL=0:GOTO2180:REM F6
2300 IFAS=140THENF2=12:GOSUB3210:F2=0:GO
TO2180:REM F8
2310 IFAS=136THENGOSUB2800: REM F7
2320 IFAS=147THENGOSUB2760: REM CLEAR
2330 IFAS=138THENGOSUB3060: REM F4
2340 IFAS=32THENGOSUB3000: REM WISSEN(SP
ACE)
2350 IFAS=13THENGOSUB2970: REM RETURN
2370 IFAS<128ANDAS>32OR(JOAND16)=0THENFL
=1:GOSUB3020:FL=0
2380 POKEO+4,SH:POKEO+5,SV:GOTO2180
2450 REM *** UP ***
2460 IFSV>106THENSV=SV-8:BV=BV-40
2470 RETURN
2480 REM *** DOWN ***
2490 IFSV<106+7*8THENSV=SV+8:BV=BV+40
2500 RETURN
```

```
2510 REM *** RIGHT ***
2520 IFSH<73+7*8THENSH=SH+8:BH=BH+1
2530 RETURN
2540 REM *** LEFT ***
2550 IFSH>73THENSH=SH-8:BH=BH-1
2560 RETURN
2570 REM *** GEGEVENS UITVOEREN (F1) ***
2580 FORVV=0TO7:K(VV)=0:NEXT:GOSUB3130
2590 POKE646,12:PRINT"□▣               MANI
PULATIES                 "
2600 PRINT:PRINT:PRINT:PRINT"▣ 1 ■  GEGE
V. UITVOEREN":PRINT
2610 PRINT"▣ 2 ■  TEKENS VERANDEREN":PRI
NT:PRINT"▣ 3 ■  TEKENS OPSLAAN":PRINT
2620 PRINT:PRINT"▣ 4 ■  MENU":POKE198,0
2621 PRINT:PRINT"▣ 5 ■  EDITOR"
2630 GETS$:IFS$=""THEN2630
2633 IFS$="5"THENGOTO530
2640 IFS$="4"THENPOKE51000,1:RUN
2650 IFS$="1"THEN2700
2655 IFS$="2"THENKL$="▣INVOER VAN NIEUW
TEKEN":CH=1:GOSUB720:CH=0:GOTO2590
2660 IFS$="3"THENGOSUB3380:CLR:O=53248:G
OTO2590
2700 PRINT"□▣▣           GEGEVENS UITVOERE
N                 "
2710 POKE214,4:PRINT:PRINT"▣DE GEGEVENS
VAN HET TEKEN ZIJN:":PRINT
2720 FORVV=0TO7:PRINTVVTAB(12)K(VV)"=":N
EXT:FL=12:F4=5:F3=1:GOSUB3210:FL=0:F3=0
2722 PRINT:POKE646,6:PRINT"DRUK EEN TOET
S !"
2730 GETF$:IFF$=""THEN2730
2731 GOTO2590
2760 REM *** USER CLEAR (CLEAR-TOETS) **
*
2770 C2=CO:CO=32:F2=12:FL=1:GOSUB3210:F2
=0:CO=C2:FL=0:RETURN
2780 REM *** MATRIX RECHTS (F5) ***
2790 FL=1:GOTO3210
2800 REM *** USED RECHTS (F7) ***
2810 FL=0:GOTO3210
2820 REM *** TEKEN  TESTEN (F3) ***
2830 H=0:FORVV=0TO7:K(VV)=0:NEXT:GOSUB31
30
2840 POKE199,1:POKE646,12:PRINT"□
        TEST                 "
2850 POKEO+21,3:POKEO,100:POKEO+1,100
2860 POKE2040,13:POKE2041,13:POKEO+39,6:
```

```
POKEO+40,6
2870 POKEO+2,192:POKEO+3,94:POKEO+29,2:P
OKEO+23,2
2880 FORVV=0TO7:POKE832+ZA,K(VV):ZA=ZA+3
:NEXT:ZA=0
2890 POKE214,7:PRINT:PRINTTAB(8)"1:1";TA
B(20)"1:2"
2891 POKE214,15:PRINT:PRINT"MOET HET TEK
EN OPGESLAGEN WORDEN ?▮▮▮(J/N)"
2900 GETA$:IFA$=""THEN2900
2910 IFA$<>"J"THENPOKEO+21,0:GOSUB1970:F
L=12:F2=12:GOSUB3210:FL=0:F2=0:GOTO2150
2920 POKEO+21,0:GOSUB3380
2930 CLR:O=53248:PRINT"TERUG NAAR MENU(J
/N) ?"
2940 GETA$:IFA$=""THEN2940
2950 IFA$="J"THENGOTO360
2960 GOTO530
2970 REM *** RETURN-TOETS ***
2980 IFSV<106+7*8THENSH=73:SV=SV+8:BH=0:
BV=BV+40:RETURN
2990 SH=73:BH=0:RETURN
3000 REM *** SPATIEBALK (WISSEN) ***
3010 POKE1024+6+7*40+BV+BH,32:RETURN
3020 REM *** PUNT PLAATSEN ***
3030 POKE1024+6+7*40+BV+BH,81:POKE55296+
6+7*40+BV+BH,3:RETURN
3050 :
3060 REM *** TEKEN INVERTEREN ***
3070 POKEO+41,14
3080 FORK=0TO7:FORKK=0TO7:PE=PEEK(1024+2
86+KK+40*K):IFPE=32THENPE=81:GOTO3100
3090 IFPE=81THENPE=32
3100 IQ=286+KK+40*K:POKEIQ+1024,PE:POKEI
Q+55296,7:NEXTKK:NEXTK:POKEO+41,3
3110 RETURN
3120 :
3130 REM *** DATA BEREKENEN ***
3140 POKE959,52:POKE961,234:SYS957:POKEO
+21,0:H=0:REM INTERRUPT & SPRITE OFF
3150 FORVV=0TO7:FORV=0TO7
3160 A(V)=PEEK(1024+H+293-V):M=2↑V:IFA(V
)=81THENK(VV)=K(VV)+M
3180 NEXTV:PRINTK(VV)"▮";:V=0:H=H+40:NEX
TVV:H=0:RETURN
3200 :
3210 REM *** TEKEN VERGROTEN ***
3220 POKEO+41,14:JJ=0:DD=293:HK=32:IFF3=
1THENHK=87
```

```
3230 IFF2=12THENDD=DD-13
3240 IFFL=1THENCG=13*4096:FA=5:GOTO3260
3250 CG=28672:FA=13
3260 IFF4=0THENF4=FA
3270 FORI=0TO7:REM REGELTELLER
3280 IFFL=1THENPOKE56334,0:POKE1,51:REM
KARAKTERGENERATOR UITLEZEN
3290 IFFL=12THENZZ=K(I):GOTO3310
3300 ZZ=PEEK(CG+8*CO+I)
3310 IFFL=1THENPOKE1,55:POKE56334,1
3311 POKE1028+6*40+JJ,32:POKE55300+7*40+
JJ,FA:POKE1028+7*40+JJ,31
3320 FORJ=0TO7:AK=ZZAND2↑J:GETST$
3322 IFST$<>""THENIFASC(ST$)=3THENPOKE10
28+7*40+JJ,32:POKEO+41,3:RETURN
3330 IFAKTHENPOKE1024+13+DD-J+JJ,81:POKE
55296+13+DD-J+JJ,F4:GOTO3350
3340 POKE1024+13+DD-J+JJ,HK:POKE55296+13
+DD-J+JJ,FA
3350 NEXT:JJ=JJ+40:NEXTI:FL=0:F4=0:POKEO
+41,3:POKE1028+6*40+JJ,32:RETURN
3370 :
3380 REM *** TEKEN OPSLAAN ***
3382 AU=PEEK(53246):FORVV=0TO7:POKE28672
+8*CO+VV,K(VV)
3383 IFCO>383ORCO>127ANDCO<256THENRV=1
3384 IFAU=1THENIFRV=1THENPOKE28672+(8*(C
O-128))+VV,255-K(VV):GOTO3389
3386 IFAU=1THENPOKE28672+(8*(CO+128))+VV
,255-K(VV)
3389 NEXT:RETURN
3390 :
3391 REM *** PROGRAMMAGENERATOR ***
3392 PRINT"♥M O M E N T !":WK=159:POKE53
000,0:POKE53001,0
3393 DIMWH(255):DIMWI(255):FORA=0TO255:W
H(A)=-1:WI(A)=-1:NEXT:MD=189
3394 CH=1:GOSUB810:CH=0
3396 POKE27648+512,WK:POKE55296+512,13:P
OKE27648+514,141:POKE55296+514,2
3397 POKE27648+516,144:POKE55296+516,2:G
OTO4000
3399 XY=PEEK(53000):YX=PEEK(53001):IFCO>
255THENGOTO3405
3400 FORG=0TO7:K(G)=PEEK(28672+8*CO+G):N
EXTG:IFWH(CO)>-1THEN3402
3401 WH(CO)=20000+XY*9:XY=XY+1:POKE53000
,XY:IFXY+YX>255THENGOTO3409
3402 POKEWH(CO),CO:FORYO=0TO7:POKEWH(CO)
```

```
+1+YO,K(YO):NEXTYO:RETURN
3405 FORG=0TO7:K(G)=PEEK(28672+8*CO+G):N
EXTG:KO=CO-256:IFWI(KO)>1THEN3407
3406 WI(KO)=22049+YX*9:YX=YX+1:POKE53001
,YX:IFXY+YX>255THENGOTO3409
3407 POKEWI(KO),KO:FORI=0TO7:POKEWI(KO)+
1+I,K(I):NEXTI:RETURN
3409 POKE53002,0:POKE53100,PEEK(53000):P
OKE53200,PEEK(53001)
3410 ZK=PEEK(53000):ZK=ZK-1:IFZK<0THEN34
60
3420 POKE53000,ZK:CO=PEEK(20000+9*ZK):FO
RI=0TO7:K(I)=PEEK(20000+9*ZK+I+1):NEXTI
3430 PRINT"{CLR}{DOWN}{DOWN}{DOWN}":ZC=PEEK(53002):POKE5300
2,ZC+1
3440 PRINTZC+370"DATA"CO"{LEFT},";:FORI=0TO6:
PRINTK(I)"{LEFT},";:NEXTI:PRINTK(7)
3450 PRINT"GOTO3410":FORM=0TO9:POKE631+M
,13:NEXTM:POKE198,10:PRINT"{HOME}":END
3460 ZL=PEEK(53001):ZL=ZL-1:IFZL<0THEN35
10
3470 POKE53001,ZL:CO=PEEK(22049+9*ZL):FO
RI=0TO7:K(I)=PEEK(22049+9*ZL+I+1):NEXTI
3480 PRINT"{CLR}{DOWN}{DOWN}{DOWN}":ZC=PEEK(53002):POKE5300
2,ZC+1
3490 PRINTZC+370"DATA"CO+256"{LEFT},";:FORI=0
TO6:PRINTK(I)"{LEFT},";:NEXT:PRINTK(7)
3500 PRINT"GOTO3460":GOTO3525
3510 ZC=PEEK(53002)+370:PRINT"{CLR}{DOWN}{DOWN}{DOWN}"ZC+1"
DATA-1":PRINTZC+2":::"
3511 PRINT"340 READA:IFA=-1THEN360"
3520 PRINT"350 FORK=0TO7:READB:POKE28672
+A*8+K,B:NEXTK:GOTO340"
3521 PRINT"360 POKE53272,189:POKE56576,1
50:POKE648,108:?CHR$(147)"
3522 PRINT"GOTO3530"
3525 FORM=0TO9:POKE631+M,13:NEXTM:POKE19
8,10:PRINT"{HOME}":END
3530 POKE53002,4:POKE53010,0
3540 AA=PEEK(53002):IFAA=0THENGOTO3620
3550 POKE53002,AA-1
3560 PRINT"{CLR}{DOWN}{DOWN}{DOWN}":ZD=PEEK(53010):FORA=0TO
7:PRINTZD+10*A:NEXTA:POKE53010,ZD+6*10
3570 PRINT"GOTO3540":GOTO3525
3620 KK=PEEK(43)+256*PEEK(44)+500+(PEEK(
53100)+PEEK(53200)*10)
```

```
3622 FORJ=KKTOPEEK(45)+256*PEEK(46)
3630 IFPEEK(J)=58ANDPEEK(J+1)=58ANDPEEK(
J+2)=58THENGOTO3650
3640 NEXTJ
3650 VH=INT((J/256)+1)
3655 PRINT"]▮▮":PRINT"POKEJ-3,0:POKEJ-4,
0:POKE45,0:POKE46,"VH":CLR":GOTO3525
3699 :
3700 REM *** ROTATE ***
3710 INPUT"HOEVEEL RECHTSROTATIES (1/2/3
) ";A$
3720 POKE781,23:SYS59903:PRINT"]":FORA=0
TO7:K(A)=0:W(A)=0:NEXTA
3730 D=VAL(A$):IFD=0THENGOTO2150
3731 GOSUB3130:POKE781,24:SYS59903:IFD=<
1ORD>3THEND=1
3732 POKEO+41,14:POKE959,144:POKE961,3:S
YS957:FORC=1TOD:FORA=0TO7:W(A)=0:NEXTA
3825 FORB=0TO7:FORA=7TO0STEP-1:M=2↑A:IFK
(B)-M>=0THENW(A)=W(A)+2↑(B):K(B)=K(B)-M
3830 NEXTA:NEXTB:FORA=0TO7:K(A)=W(A):NEX
TA:FORA=0TO7:A(A)=K(A):NEXTA
3840 FORA=0TO7:K(A)=A(7-A):NEXTA:NEXTC
3850 F2=12:FL=12:GOSUB3210:PRINT"]T]":F2=
0:FL=0:POKEO+41,3:GOTO2150
4000 FR=14:LE=0:R=0:IFKR=255THENPRINTCHR
$(14)
4001 GETC$:JO=PEEK(56320):IFC$=""THENC$=
CHR$(0)
4002 C=ASC(C$):IFC=157OR(JOAND4)=0THENIF
LE>0THENR=-2:GOSUB4050
4004 IFC=29OR(JOAND8)=0THENIFLE<516THENR
=2:GOSUB4050
4006 IFC=13OR(JOAND16)=0THENCO=PEEK(2764
8+LE)+KR:FR=7:R=0:GOSUB4050:GOSUB4060
4007 IFFL=3THENPOKE55296+LE,1
4008 IFFL=6THENFL=0:POKE55296+LE,FR
4009 FL=FL+1
4010 IFC=145OR(JOAND1)=0THENIFLE>=40THEN
R=-40:GOSUB4050
4015 IFC=17OR(JOAND2)=0THENIFLE<=11*40+3
6THENR=+40:GOSUB4050
4030 GOTO4001
4050 FL=3:POKE55296+LE,FR:LE=LE+R:FR=PEE
K(55296+LE):RETURN
4060 IFLE=512THENIFPEEK(53272)<>191THENK
R=256:WK=32:MD=191:GOTO3394
4062 IFLE=514THENPRINTCHR$(142):POKE5327
2,21:POKE56576,151:POKE648,4:POKE51000,1
```

```
4064 IFLE=516THENGOSUB4070:GOTO3409
4065 GOSUB3399:RETURN
4070 POKE53272,21:POKE56576,151:POKE648,
4
4080 PRINT"♥":POKE214,10:PRINT:PRINT"DE
EDITOR WORDT HIERDOOR GEWIST...!"
4081 PRINT"(J/N) ??!"
4090 GETK$:IFK$=""THEN4090
4095 IFK$="J"THENRETURN
4096 GOTO 360
```

Dankzij het menu vindt u vanzelf de weg in het programma. Nog enkele opmerkingen tot slot.

AUTO definieert automatisch de reverse tegenhanger van een teken. Na het oproepen van de programmagenerator wordt de eerste karakterset gelist. Ga via toetsenbord of joystick naar de tekens, die deel van de BASIC-lader moeten gaan uitmaken en druk daarna op RETURN. Een teken verandert van kleur als het is ingevoerd. Aan de onderkant kunnen rechts drie functies worden gekozen: (1) pijl naar links: listen van de tweede karakterset; (2) reverse M: terug naar menu bij foute invoer; (3) reverse P: programmagenerator activeren, die BASIC-lader levert. De nieuwe karakerset ligt aan het einde van BASIC.

5.7 DESIGN in de listing

Bij het programma in deze paragraaf is het teken niet vervat in een BASIC-lader; de vorm zelf staat in de listing en wordt in bytewaarden omgerekend. Deze manier is wat langzamer, omdat de bytewaarden eerst moeten worden omgerekend. Het grote voordeel is de overzichtelijkheid en het feit, dat er door verkeerde waarden niets fout kan gaan, omdat de vorm deel uitmaakt van de listing.

```
1 PRINT"♥ EVEN GEDULD! KOPIEERROUTINE D
UURT ▒             CA. 1 MINUUT !"
10 POKE 56334,0:POKE 1,51
15 FOR K=0 TO 4095
20 POKE 57344+K,PEEK(53248+K)
25 NEXT
30 POKE 1,55:POKE 56334,1
```

```
50 POKE 53272,PEEK(53272) AND NOT 6 OR 8
60 POKE 56576,196:POKE 648,3*64+4:PRINT"
♥"
100 READ A$:IF A$<>"DESIGN" THEN GOTO 10
0
110 READ B$:B$=LEFT$(B$,1)
120 PRINT"■";B$
130 CO=PEEK(50176)
140 FOR K=0 TO 7:READ C$
141 IF C$="EINDE" THEN PRINT"? MISSING S
TATEMENTS IN";PEEK(63)+256*PEEK(64):END
150 IF LEN(C$)<>8 THEN PRINT"? TOO MANY
SIGNS IN";PEEK(63)+256*PEEK(64):END
160 FOR J=7 TO 0 STEP-1:K$=MID$(C$,8-J,1
):IF K$="*" THEN BY=BY+2↑J
170 NEXT J:POKE 57344+8*CO+K,BY:BY=0:NEX
TK
180 READ A$:IF A$="DESIGN" THEN GOTO 110
200 DATA DESIGN,A
210 DATA..***...
220 DATA.**.**..
230 DATA**...**.
240 DATA**...**.
250 DATA*******.
260 DATA**...**.
270 DATA**...**.
280 DATA**...**.
290 DATA DESIGN,B
300 DATA          *******.
310 DATA          *......*
320 DATA          *......*
330 DATA          *.....*.
340 DATA          ******..
350 DATA          *......*
360 DATA          *......*
370 DATA          *******.
380 DATA DESIGN,C
390 DATA..****..
400 DATA.*....*.
410 DATA*.......
420 DATA*.......
430 DATA*.......
440 DATA*.......
450 DATA.*....*.
460 DATA..****..
480 DATA EINDE
```

In de eerste dataregel staat de DESIGN-instructie, een tekendefinitie, gevolgd door het teken, dat moet worden veranderd. Daarna komt het nieuwe teken. Sterretjes geven de vorm aan.

5.8 MULTI COLOR

In de MULTI COLOR (MC) modus is de horizontale oplossing van de sprites de helft van de normale oplossing. Er kunnen drie kleuren tegelijkertijd worden gebruikt. Ook zelfgedefinieerde tekens kunnen MULTI COLORed worden afgebeeld. Bit 4 van register 22 (stuurregister 2) van de VIC bepaalt of de tekens in de MC-modus worden afgebeeld of niet.

```
POKE 53270, PEEK(53270) OR 16
```

brengt u in de MC-modus. Met

```
POKE 53270, PEEK(53270) AND NOT 16
```

komt u weer in de standaardmodus. In de MC-modus kunnen maar 8 in plaats van 16 kleuren worden gebruikt, de grondkleuren zwart tot geel. Tekens die een andere kleur hebben, zijn op het scherm moeilijk te herkennen. Ze zijn hoekiger en hebben bij veel kleurencombinaties een schaduw. In de MC-modus is een punt op het scherm in feite twee punten naast elkaar. Er zijn vier combinatiemogelijkheden:

```
00 = kleur 0: achtergrondkleurregister 0 (adres 53281)
01 = kleur 1: achtergrondkleurregister 1 (adres 53282)
10 = kleur 2: achtergrondkleurregister 2 (adres 53283)
11 = kleur 3: tekenkleur                 (adres 646)
```

We moeten nu iets preciezer worden. Alleen de kleur van letters is in de MC-modus beperkt tot de acht genoemde grondkleuren. Behalve letters kunnen acht MC-tekens worden uitgevoerd. Het kleur-RAM heeft 8 verschillende kleuren, andere registers hebben er 16. In het volgende programma wordt de DESIGN-instructie uitgebreid tot DESIGN-MC. Daarmee kunnen MULTI COLORed tekens worden gemaakt, bijvoorbeeld:

```
300 DESIGN-MC,B                     1=kleur-RAM
310 DATA 1111                       2=kleurregister 2
320 DATA 2..2                       3=kleurregister 1
330 DATA 2..2
340 DATA 3..3
350 DATA 3..3
360 DATA 1111
370 DATA EINDE
```

De listing:

```
1 PRINT"♥ CA. 1 MINUUT WACHTEN!":POKE646
16
10 POKE 56334,0:POKE 1,51
15 FOR K=0 TO 4095
20 POKE 57344+K,PEEK(53248+K)
25 NEXT
30 POKE 1,55:POKE 56334,1
50 POKE 53272,PEEK(53272) AND NOT 6 OR 8
60 POKE 56576,196:POKE 648,3*64+4:PRINT"
♥"
100 READ A$:IF LEFT$(A$,6)<>"DESIGN"THEN
 GOTO 100
101 LL=8:MM=7
102 IF A$="DESIGN-MC"THEN MC=1:LL=4:MM=3
110 READ B$:B$=LEFT$(B$,1)
120 PRINT"■";B$
130 CO=PEEK(50176)
140 FOR K=0 TO 7:READ C$
141 IF C$="EINDE"THEN PRINT"? MISSING ST
ATEMENTS IN";PEEK(63)+256*PEEK(64):END
150 IF LEN(C$)<>LL THEN PRINT"? TOO MANY
 SIGNS IN";PEEK(63)+256*PEEK(64):END
160 FOR J=MM TO 0 STEP-1:K$=MID$(C$,LL-J
,1):IF K$="*"THEN BY=BY+2↑J
161 IF MC=0 THEN GOTO 170
162 IF K$="1" THEN BY=BY+2↑(2*J):BY=BY+2
↑(2*J+1)
163 IF K$="2"THEN BY=BY+2↑(2*J)
164 IF K$="3"THEN BY=BY+2↑(2*J+1)
170 NEXT J:POKE 57344+8*CO+K,BY:BY=0:NEX
T K
180 READ A$:IF LEFT$(A$,6)="DESIGN"THEN
GOTO 101
190 POKE 53270,PEEK(53270)OR 16:POKE 646
,8:PRINT"BBBB":REM MC ON
200 DATA DESIGN,A
210 DATA...*....
```

```
220 DATA.**.**..
230 DATA**...**.
240 DATA**...**.
250 DATA*******.
260 DATA**...**.
270 DATA**...**.
280 DATA**...**.
290 DATA DESIGN-MC,B
300 DATA            1111
310 DATA            2222
320 DATA            3333
330 DATA            ....
340 DATA            3333
350 DATA            2222
360 DATA            1111
370 DATA            ....
390 DATA EINDE
```

5.9 Transparanten printen zonder moeite

Voer de tekst in en kies de vergrotingsfactor. Die van de breedte is vrij, die van de hoogte mag maximaal 10 zijn. De ingevoerde tekst wordt teken voor teken ingelezen. Als de vorm uit het ROM is vergroot, wordt het teken gedraaid. De listing spreekt voor zich.

```
5 PRINT"□":POKE53280,14:POKE53281,14:POK
E646,1
10 REM *** TRANSPARANT-PRINTER ***
15 PRINT"■      TRANSPARANT-PRINTER
■■■■"
16 PRINT"PRINTER INSCHAKELEN !■■■■"
20 OPEN4,4
21 INPUT"BREEDTE";BR
22 INPUT"HOOGTE";HO
23 PRINT"□"
25 POKE 214,21:POKE 211,VV:SYS 58732:POK
E 204,0
26 GET T$:IF T$="" THEN 26
27 IF PEEK(207)THEN 27
28 POKE 204,1:IF T$=CHR$(133) THEN PRINT
"□O.K.":END
```

```
29 K$=K$+T$:IF LEN(K$)>39 THEN K$=RIGHT$
(K$,(LEN(K$)-1))
30 POKE 214,20:PRINT:PRINT K$:IF VV<40 T
HEN VV=VV+1
31 TX=0:IFPEEK(53272)=23THENTX=256
51 POKE646,14:PRINT"▒";T$:CO=PEEK(1024):
POKE646,1
60 REM *** TEKEN INLEZEN ***
70 POKE 56334,0:POKE 1,51
80 FOR A=0 TO 7
90 CH(A)=PEEK(53248+(8*CO)+A+(TX*8))
100 NEXT A
110 POKE 1,55:POKE 56334,1
120 REM *** TEKEN DRAAIEN ***
130 FORA=0TO7:K(A)=0:NEXT
140 FORA=0TO7
150 FORB=7TO0STEP-1:W=2↑B:IFCH(A)>=WTHEN
CH(A)=CH(A)-W:K(7-B)=K(7-B)+2↑A
160 NEXTB,A
200 REM *** TEKEN UITVOEREN ***
210 FOR I=0 TO 7:Q$="":FOR J=7 TO 0 STEP
-1:WI=K(I) AND 2↑J
220 IF WI THEN FOR U=1 TO HO:Q$=Q$+"●":N
EXT U:GOTO 250
230 FOR U=1 TO HO:Q$=Q$+" ":NEXT U
250 NEXT J:XY$="":FOR U=1 TO BR:XY$=XY$+
Q$:NEXT U
255 LX=LEN(XY$)-1:IF RIGHT$(XY$,1)=" "TH
EN XY$=LEFT$(XY$,LX):GOTO 255
260 PRINT#4,XY$:NEXT I:GOTO 25
270 END
```

5.10 Een grote lichtkrant

Met de volgende listing kan een tekst, die in TX$ is opgeslagen, als lichtkrant worden uitgevoerd. De tekens zijn dan 8x vergroot.

```
5 POKE 53280,1:POKE 53281,1:POKE 646,6:P
RINT"♥"
10 INPUT"TEKST";TX$:TX$=TX$+"          "
20 PRINT"♥":FOR Z=1 TO LEN(TX$)
```

```
30 A$=MID$(TX$,Z,1)
40 GOSUB 1000:NEXT Z
50 END
60 :
70 :
80 :
90 :
100 :
1000 POKE 646,1:PRINT"♥";A$:CO=PEEK(1024
):POKE 646,6
1010 POKE 56334,0:POKE 1,51
1020 FOR J=0TO7
1030 CO(J)=PEEK(53248+(8*CO)+J)
1040 NEXT J
1050 POKE 1,55:POKE 56334,1
1060 FOR X=7 TO 0 STEP-1
1070 FOR Y=0 TO 7
1080 A=CO(Y) AND 2↑X:IF A THEN W$(Y)=W$(
Y)+"o":GOTO 2000
1090 W$(Y)=W$(Y)+" "
2000 NEXT Y:P=P+1
4000 REM
4010 PRINT"♥";:FOR W=0 TO 7
4020 IF Q>37 THEN Q=38
4021 PRINTTAB(38-Q)W$(W)
4030 NEXT W:Q=Q+1
4040 IF P<39 THEN 4100
4050 FOR T=0 TO 7
4060 LE=LEN(W$(T))-1:W$(T)=RIGHT$(W$(T),
LE)
4070 NEXT T
4100 NEXT X:RETURN
```

In BASIC is dit programma erg langzaam, maar in machinetaal valt dat mee.

5.11 Acht blocks voor sprites

Sprites kunnen alleen in de volgende vier blocks worden opgeslagen:

block 11 (704- 766)

block 13 (832- 894)
block 14 (896- 958)
block 15 (960-1020)

13-15 liggen in de cassettebuffer, die tijdens het programma meestal niet wordt gebruikt. Met deze blocks kunnen vier sprites worden gedefinieerd, maar niet alle acht. Door de BASIC-start te verplaatsen ruimt u dit probleem uit de weg en bent u niet langer afhankelijk van de cassettebuffer. In het voorbeeld worden er acht blocks (32-39) vrij. Verschuif de BASIC-start van 2048 naar 2560:

POKE 44,10	adres 43 van de vector blijft onveranderd, het beginadres van BASIC wordt 10*256
POKE 10*256,0	0 aan begin van BASIC
NEW	wijzer zetten

Deze drie instructies moeten voorafgaan aan het laden van het eigenlijke programma.

6 HET SPEL

Als u pas begint, is het spel ELEVATOR BOY heel geschikt om te leren hoe een lang spel kan worden geprogrammeerd. Gevorderden kunnen er misschien nog wat kneepjes bij leren. Als piccolo in een hotel haalt u schoenen op, die gepoetst moeten worden. Per paar geven de gasten een gulden fooi. Als u meer dan 50 schoenen bij de portier aflevert, verdient u een hoog bedrag extra. Maar als u verkeerd in de lift springt, valt u in de schacht en verliest alle schoenen. Gebruik de joystick aan poort 2. Een druk op de knop van de joystick verandert de richting van de lift. De eerste keer is het heel lastig om goed in de lift te komen, dat wil zeggen om de joystick op het juiste moment in de goede richting te duwen. Terwijl u dat probeert, wordt de bonus in de vorm van een verticale streep steeds kleiner. Voordat de bonus op is, moet u de portier hebben bereikt. Daarna verschijnt een nieuw beeld en een nieuwe bonus. Bent u niet op tijd bij de portier, dan kost dat een leven.

In de eerste listing is de grafiek heel slecht. Dat wordt beter, als in de tweede stap de tekens opnieuw worden gedefinieerd. Omdat u in dit stadium waarschijnlijk nog geen wijs kunt worden uit de tekens, laten we zien wat u later mag verwachten:

TEKEN	BETEKENIS
/pond/	kamer met een paar schoenen
/pijl omhoog/	gesloten deur
#	speler
%	portier
"	lift

De volgorde van de uitbreidingen geeft tevens het belang van de punten aan:

0 - geraamte
1 - grafiek
2 - sound
3 - gebruiksaanwijzing
4 - beginbeeld

6.1 Het geraamte

In het geraamte is voldoende ruimte open gelaten voor grafiek en dergelijke. Tussen 6-12 (naam en definitie van variabelen) en 100 komt het beginbeeld te staan. Met de listing in deze paragraaf kan het spel nog niet worden gespeeld. Start het programma wel, zodat u kunt zien hoe het spel zich ontwikkelt. In paragraaf 6.6 staat de listing van het hele spel.

```
6 PRINT"██":POKE53281,3
7 BBB=1024:R=54272
9 X=BBB+536:D=BBB+487:G=40:BO=23:PUN=0
10 FE=10:LI=3
11 PRINT"█████████████                 PICCOLO
    "
12 IFPEEK(56320)<>111 THEN12
100 BEELD$(1)=" !↑↑£££ ↑↑£££ ↑↑££↑ £££££
 £££££!"
110 BEELD$(2)=" !↑↑↑↑£ £££££ £↑£££ ££££↑
 £££££!"
120 BEELD$(3)=" !↑££££ ↑↑↑££ £££££ £££££
 ££££↑!"
130 BEELD$(4)=" !£££££ £££££ £↑£££ ££↑↑£
 ££↑↑↑!"
140 BEELD$(5)=" !↑££££ £££££ £££££ £££££
 ↑↑↑↑↑!"
150 BEELD$(6)=" !£££££ ££↑↑↑ ££££↑ ££££↑
 ££££↑!"
160 BEELD$(7)=" !£££££ £↑↑££ ££↑££ ££££↑
 £££↑↑!"
170 BEELD$(8)=" !£££££ ↑↑£££ £££££ ££↑££
 ££↑↑↑!"
180 BEELD$(9)=" !£££££ ↑↑£££ £££££ ↑↑↑↑£
 £££££!"
183 PRINT"█";
184 PRINT" █←←←←←←←←←←←←←←←←←←←←←←←←←←←←
←←←"
185 PRINT" █←←←←←←←←←←←←←←←←←←←←←←←←←←←←
←←←"
187 PRINT" █!█←←←←← ←←←←← ←←←←← ←←←←← ←←
←←←█!"
190 FORI=1TO9
195 W=INT(RND(1)*8)+1
200 PRINTBEELD$(W)
205 PRINT" █!█←←←←← ←←←←← ←←←←← ←←←←← ←←
```

```
←←←▒!"
210 NEXT
215 PRINT"▒ !                    %
  !"
217 PRINT"▒ !█←←←←← ←←←←← ←←←←← ←←←←← ←←
←←←▒!"
220 PRINT"█ ←←←←←←←←←←←←←←←←←←←←←←←←←←←←
←←←"
221 FORI=BBB+77TOBBB+917STEP40:POKEI,38:
POKEI+R,6:NEXT
230 POKEX,35:POKEX+R,0:POKEX+1,32:POKEX+
2,32:POKEX-1,32:POKEX-2,32
270 PRINT"          HI-SCORE ";HIS;
280 PRINT"TTTTTTTTTTTTTTTTTTTTTTTTTTTTTT"
1000 GOSUB3000
1003 FORI=1TOFE:NEXT
1005 POKED,32:POKED+6,32:POKED+12,32:POK
ED+18,32
1010 IFPEEK(D+G)=31THENG=-G
1020 D=D+G
1030 POKED,34:POKED+6,34:POKED+12,34:POK
ED+18,34
1040 IFOL=0THEN2000
1050 POKEX,32:X=X+G
1060 IFPEEK(X)=31THENX=X+40:GOTO9000
1070 POKEX,35:POKEX+R,0
2000 BO=BO-0.025
2010 POKEINT(BO)*40+BBB+37,32:IFBO=<1THE
N9000
2990 GOTO1000
3000 IFPEEK(56320)=127THENH=0:RETURN
3010 IFPEEK(56320)=<111THEN3500
3020 IFPEEK(56320)=123THENY=-1:GOTO3050
3030 IFPEEK(56320)=119THENY=1:GOTO3050
3040 RETURN
3050 IFPEEK(X+Y)=31THENY=Y-40:IFPEEK(X+Y
)=31THEN9000
3053 IFPEEK(X+Y)=28THENPUN=PUN+1:GOTO310
0
3054 IFPEEK(X+Y)=33THENRETURN
3055 IFPEEK(X+Y)=30THENRETURN
3057 IFPEEK(X+Y)=34ORPEEK(X+Y+40)=34THEN
3900
3060 IFPEEK(X+Y)=37THEN3800
3065 IFPEEK(X+Y)=32ANDPEEK(X+Y+40)=32THE
N:POKEX,32:X=X+Y:GOTO9000
3070 IFPEEK(X+Y)=32ANDPEEK(X+Y+40)=102TH
EN3100
3100 POKEX,32
```

```
3110 X=X+Y:OL=0
3120 POKEX,35:POKEX+R,0
3130 PRINT"] ←PUNTEN←";PUN+TR;:PRINT"←←S
CHOENEN←";PUN
3190 RETURN
3500 IFH=0THENG=-G:H=1
3510 RETURN
3800 FE=FE-1:IFPUN<50THENTR=TR+PUN:GOTO3
810
3805 TR=TR+INT(PUN*(BO*7))
3810 X=BBB+536:D=BBB+487:G=40:BO=23:PUN=
0
3830 GOTO183
3900 POKEX,32
3905 X=X+Y
3910 IFPEEK(X)=34THENX=X-40:IFPEEK(X)=34
THEN9000
3920 OL=1
3930 POKEX,35:POKEX+R,0
3940 RETURN
9000 TR=TR+PUN:LI=LI-1
9010 IFBO=<1THENGOTO9020
9015 POKEX,32:X=X+40:IFPEEK(X+40)=31THEN
9100
9017 POKEX,35:POKEX+R,0:GOTO9015
9020 POKEX,32
9030 BO=23:FORI=1TO23:POKEI*40+BBB+37,16
0:NEXT:OL=0
9035 D=BBB+487:G=40:BO=23:PUN=0
9036 IFLI=0THEN9800
9040 X=BBB+536:POKEX,35:POKEX+R,0:GOTO18
3
9100 POKEX,39:POKEX+R,0
9190 GOTO9020
9800 PRINT"]";:POKE53281,1
9830 PRINT"QQQQQQQQQQQQ                JE HEBT
 ";TR+PUN;"$ VERDIEND"
9840 IFTR+PUN>HISTHENHIS=TR+PUN
9850 TR=0:PUN=0
9880 GETA$:IFA$<>""ORPEEK(56320)=111THEN
6
9890 GOTO9880
```

Voor een snel overzicht over de variabelen moeten ze in de eerste regels worden gedefinieerd. Tabel van de belangrijkste variabelen in volgorde van belangrijkheid:

X	Adres, waar de speler zich bevindt. X is in het *begin 1560, ongeveer het midden van het scherm.*

BBB	Adres van het begin van het beeldschermgeheugen. Deze variabele moet worden veranderd, als de karakterset wordt veranderd. De variabele dient hier als wijzer en is een constante.
R	R komt overeen met BBB en is het verschil tussen het begin van het beeldschermgeheugen en het begin van het kleurgeheugen. Deze variabele (constante) moet worden veranderd bij het gebruik van nieuwe grafische tekens. Bovendien vereenvoudigt ze de verandering van de achtergrondkleur van een teken. POKE X+R,(kleur van de speler) verandert de achtergrondkleur van de speler.
D	Adres van de eerste liftvloer. De tweede is D+6, de derde D+12 enzovoort. De verhoging van de variabele bespaart programmaregels.
BONUS	Variabele voor de bonus. Zorg er bij lange spelen voor, dat u aan de naam van de variabele de betekenis kunt herkennen.
Y	Hulpvariabele voor de coördinaten van de speler. Er wordt bijvoorbeeld mee gecontroleerd of het volgende veld vrij is.
TR	Het verdiende loon.
PUN	Aantal schoenen, dat de speler bij zich heeft. PUN wordt 0, als u in de liftschacht valt.
LI	Aantal levens. Bij elke val of eind van een bonus verliest u een leven. Automatisch wordt nagegaan of LI<=0.
FE	Snelheid. Deze is aanvankelijk 10 en kan vrijwel niet hoger worden. Als beginner kunt u de snelheid verlagen door FE te verhogen. Bent u bij de portier, dan wordt FE met 5 verminderd.
G	Richting van de lift. 40=naar beneden, -40=naar boven. D+G is de beweging van de lift.
I	I is een lusvariabele. Het is het handigst om in verschillende programma's steeds dezelfde variabelen hiervoor te reserveren. Het bespaart geheugenruimte, als u in een programma steeds dezelfde lusvariabele gebruikt. Zorg er wel

	voor, dat een lus afgesloten is, alvorens de variabele nogmaals te gebruiken.
HIS	HIGH SCORE: het hoogste aantal punten, dat behaald werd.
H	H verhindert dat met een druk op de knop de richting van de lift meerdere keren verandert. Een druk op de vuurknop verandert de richting H=1 en wordt pas 0, als de joystick in de ruststand is. Als H=1, reageert de computer niet op de drukknop van de joystick. Op dezelfde manier kunt u bij het spel FROGGER de lengte van de beweging beperken.
OL	Als OL=1 bent u in de lift.
W	Random getal om verschillende beelden te maken.
BEELD$(W)	BEELD$(W) kan 9 verschillende variabeletoewijzingen (beelden) bevatten. Deze gedimensioneerde variabele kiest steeds een ander beeld. Hoe meer beelden, des te interessanter het programma.

Ook de opsomming van de afzonderlijke stappen is van belang. Hieronder staan de programmaregels en hun functie:

0 - 5	Op deze plaats wordt de karakterset ingelezen en geactiveerd (zie paragraaf 6.2).
6 - 99	Dit bereik is gereserveerd voor de definitie van variabelen. Bovendien bevindt zich hier het eenvoudige titelbeeld. Later staat er het verbeterde beginbeeld.
100 - 180	Definitie van variabele BILD$(W).
183 - 280	Hier wordt het totaalbeeld gemaakt.
1000 -2990	De beweging van de lift wordt geregeld.
1050 -1070	De lift in en uit. Deze beweging loopt apart van de normale beweging van het figuurtje.
3000 -3190	De besturing van het figuurtje. In tegenstelling tot 1050-1070 beïnvloedt de joystick deze beweging.
3500 -3510	Verandering van de beweging van de lift.

3800 -3830	Het programma springt naar deze regels als het figuurtje bij de portier is.
3900 -3940	Dient voor de beweging van het figuurtje.
9000 -9190	De val in de schacht wordt verwerkt.
9800 -9890	De eindroutine. Die wordt aangesproken als er geen levens meer zijn.
50000-einde	DATA-regels voor de grafiek.

In dit overzicht ontbreken de regels voor de geluidseffecten. Met een RENUMBER-programma kunt u de afstanden tussen de bereiken verwijderen.

6.2 Grafiek

De uitbreiding van de grafische bijzonderheden is de langste vanwege de vele definities van tekens en de gegevens, die daaruit volgen. Alle 58 tekens worden veranderd:

```
1 READ A:IF A=-1 THEN 3
2 POKE 57344+I,A:I=I+1:GOTO 1
3 POKE 53272,24:POKE 56576,148:POKE 648,
196
4 GOTO 6
5 POKE 53272,21:POKE 56576,151:POKE 648,
4:END
6 PRINT"[CLR]":POKE 53281,3
7 BBB=50176:R=5120
50000 DATA127,127,81,22,20,22,56,0
50001 DATA60,60,102,102,126,126,102,0:A
50002 DATA120,124,102,124,126,103,126,0:
B
50003 DATA60,126,102,96,102,126,60,0:C
50004 DATA124,126,99,99,99,126,124,0:D
50005 DATA127,126,96,124,96,126,127,0:E
50006 DATA127,126,124,96,124,120,112,0:F
50007 DATA62,121,112,115,115,63,31,0:G
50008 DATA115,115,115,127,127,115,115,0:
H
```

```
50009 DATA28,28,28,28,28,28,28,0:I
50010 DATA127,127,7,7,103,63,31,0:J
50011 DATA115,119,126,126,126,119,115,0:
K
50012 DATA112,112,112,112,112,126,127,0:
L
50013 DATA119,127,107,107,107,99,119,0:M
50014 DATA103,115,123,127,127,119,115,0:
N
50015 DATA62,127,119,119,119,127,62,0:O
50016 DATA126,127,115,127,126,112,112,0:
P
50017 DATA62,127,99,111,111,127,62,0:Q
50018 DATA126,127,115,127,124,126,119,0:
R
50019 DATA63,126,96,62,3,63,126,0:S
50020 DATA127,127,93,28,28,28,28,0:T
50021 DATA119,119,119,119,127,127,28,0:U
50022 DATA119,119,50,58,58,28,8,0:V
50023 DATA119,99,107,107,107,127,54,0:W
50024 DATA99,119,54,62,62,119,99,0:X
50025 DATA119,59,31,14,14,14,14,0:Y
50026 DATA127,63,7,28,112,126,127,0:Z
50027 DATA240,240,192,192,192,192,240,24
0:[
50028 DATA0,36,102,102,102,102,36,0:£
50029 DATA15,15,3,3,3,3,15,15:]
50030 DATA126,126,126,78,62,126,126,126:
↑
50031 DATA239,239,239,0,251,251,251,0:←
50032 DATA0,0,0,0,0,0,0,0
50033 DATA255,0,255,0,255,0,255,0:!
50034 DATA255,255,255,189,60,60,60,60:"
50035 DATA0,60,60,24,255,24,60,36:#
50036 DATA101,85,87,87,85,85,85,101:$
50037 DATA24,24,24,60,60,126,255,48:%
50038 DATA127,254,127,254,127,254,127,25
4:&
50039 DATA0,24,24,126,126,24,24,24:'
50040 DATA24,48,96,96,96,96,48,24:(
50041 DATA24,12,6,6,6,6,12,24:)
50042 DATA48,72,72,16,16,0,16,0:*
50043 DATA0,60,60,126,126,60,60,0:+
50044 DATA0,0,0,0,0,12,24,48:,
50045 DATA0,0,0,126,126,0,0,0:-
50046 DATA0,0,0,0,0,96,96,0:.
50047 DATA0,6,12,24,24,48,96,0:/
50048 DATA0,24,60,102,102,60,24,0:0
50049 DATA0,12,28,60,12,12,12,0:1
```

```
50050 DATA0,12,18,4,8,16,30,0:2
50051 DATA0,30,2,4,2,18,30,0:3
50052 DATA0,16,16,16,20,30,4,0:4
50053 DATA0,30,16,28,2,2,28,0:5
50054 DATA0,12,18,16,28,18,12,0:6
50055 DATA0,30,18,4,8,16,16,0:7
50056 DATA0,12,18,12,18,18,12,0:8
50057 DATA0,24,36,28,4,36,24,0:9
50058 DATA-1
```

Als u het programma start, moet u eerst wachten totdat de computer alle gegevens heeft ingelezen. Voor het spel zijn niet alle 59 letters nodig, maar we hebben ze allemaal afgedrukt. Op die manier kan de karakterset ook voor eigen programma's worden gebruikt. Elke DATA-regel bevat de 8 gegevens van het teken dat erachter staat. De grafiek van ELEVATOR BOY bestaat alleen uit tekens van de beeldschermcode. Daarom behandelen we geen sprites en hi-res grafiek. Is het spel gestart, komt u met

RUN 5

weer in de normale grafiek. Dat bespaart een aantal POKEs. Is per ongeluk tegelijk op RUN/STOP en RESTORE gedrukt, doe dan hetzelfde en wis daarna het scherm. Met

RUN 3

kunt u vervolgens direkt met de grafische symbolen spelen. De karakterset wordt dan niet nog een keer ingelezen. Een voorbeeld van een extra grafisch effect is het kruis bij de val in de liftschacht. Dit wordt ondersteund door een geluidseffect. Let erop, dat de kleuren op een zwart-wit televisie zichtbaar moeten zijn: rood en blauw kunnen bijvoorbeeld niet worden onderscheiden.

6.3 Geluid

Als er iets bijzonders gebeurt kunt u een geluidseffect inlassen, melodieën of tonen. In het handboek staan een paar kleine geluidsroutines. Bij ELEVATOR BOY hoort geluid als de speler een paar schoenen pakt en als hij in de liftschacht valt.

```
250 POKE 54296,15
252 POKE 54277,31
254 POKE 54278,128
256 POKE 54272,1
260 POKE 54273,1

3053 IF PEEK(X+Y)=28 THEN POKE 54276,129:
     PUN=PUN+1:GOTO 3100
3180 POKE 54276,0

9110 POKE 54277,31
9112 POKE 54278,255
9114 POKE 54272,0
9116 POKE 54273,2
9118 POKE 54276,33
9120 FOR I=1 TO 500: NEXT I: POKE 54276,0
```

De waarden in de regels 250-260 zorgen voor de klank. Als het figuurtje een paar schoenen ophaalt, wordt de golfvorm in- en uitgeschakeld (3053 en 3180). In 9110-9120 wordt het geluid gemaakt voor de val in de schacht. In hoofdstuk 7 staat een programma voor een melodie (muziek via de interrupt). De geluidseffecten lopen via de eerste stem, de melodie gebruikt de derde.

6.4 De gebruiksaanwijzing

Het handigste is om een gebruiksaanwijzing aan het spel vooraf te laten gaan:

```
25 PRINT" !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!"
26 PRINT"WILT U INFORMATIE (J/N)*TTTT"
28 GET G$:IF G$="N"OR PEEK(56320)=111 TH
EN 100
29 IF G$="J" THEN 42
38 GOTO 28
42 PRINT"]U BENT PICCOLO IN EEN GROOT HO
TEL EN ";
43 PRINT"U  MOET SCHOENEN VERZAMELEN IN
DE              HOTELKAMERS"
44 PRINT"VOOR ELK PAAR ONTVANGT U 1 DOLL
```

```
AR"
45 PRINT"FOOI VAN DE GAST."
46 PRINT"ALS U MEER DAN 50 PAAR BIJ DE P
ORTIER"
47 PRINT"AFLEVERT, KRIJGT U EEN EXTRA DE
EL.";
49 PRINT"M              [£] KAMER MET EEN PA
AR SCHOENEN"
50 PRINT"MMM   [↑] GESLOTEN DEUR"
51 PRINT"MMM   [ ] LEGE KAMER"
52 PRINT"MMM   [#] PICCOLO"
53 PRINT"MMM   [%] PORTIER"
54 PRINT"MMM   ["CHR$(34)"]LIFT"
56 IF PEEK (56320) <>111 THEN 56"
```

Wis regel 12. De bediening van een programma moet zo gemakkelijk mogelijk zijn. Daarom hoeft u in dit spel op de vraag of u instructies wilt niet te antwoorden; een druk op de knop van de joystick is voldoende.

6.5 Beginbeeld

Het eerste beeld moet aantrekkelijk zijn. Dat verlangt een uitgekiende grafische vormgeving. Ons beginbeeld is een bewegende scene uit het spel. ELEVATOR BOY verschijnt als lichtkrant:

```
11 EL$="            PICCOLO              PICC
OLO     "
27 PRINT"?"EL$
30 FOR O=0 TO 150: NEXT
31 EL$= RIGHT$(EL$,1) + LEFT$(EL$,LEN(EL
$)-1
39 GOTO 27
```

Dit programmaatje kan ook voor een andere tekst worden gebruikt. In regel 30 staat de snelheid. Als deze regel wordt weggelaten of de snelheid 0 is, loopt de tekst zo snel, dat hij niet meer kan worden gelezen. Door LEFT$ en RIGHT$ om te draaien, verandert de richting. De volgende regels voorzien het spel van een bewegend beginbeeld:

```
12 PRINT:PRINTTAB(14)"!←←←←← ←←←←←!"
13 PRINTTAB(14)"!↑↑↑££ £££££!"
14 PRINTTAB(14)"!←←←←← ←←←←←!"
15 PRINTTAB(14)"!↑££££ £↑↑↑↑!"
16 PRINTTAB(14)"!←←←←← ←←←←←!"
17 PRINTTAB(14)"!£££££ £££££!"
18 PRINTTAB(14)"!←←←←← ←←←←←!"
19 PRINTTAB(14)"!↑↑£££ £££££↑!"
20 PRINTTAB(14)"!←←←←← ←←←←←!"
21 PRINTTAB(14)"!↑↑↑↑£ £££↑↑!"
22 PRINTTAB(14)"!←←←←← ←←←←←!"
23 PRINT:PRINTTAB(14)"!←←←←← ←←←←←!"
32 PL=BBB+100+M
33 POKE PL-40,32
34 POKE PL,35:POKE PL+R,0
35 POKE PL+40,34:POKE PL+R+40,2
36 POKE PL+80,0
37 M=M+G:IF M>400 OR M<=0 THEN G=-G
```

6.6 Samenvatting

De volledige listing van het programma is als volgt:

```
1 READ A:IF A=-1 THEN 3
2 POKE 57344+I,A:I=I+1:GOTO 1
3 POKE 53272,24:POKE 56576,148:POKE 648,
196
4 GOTO 6
5 POKE 53272,21:POKE 56576,151:POKE 648,
4:END
6 PRINT"[illegible]":POKE 53281,3
7 BBB=50176:R=5120
9 X=BBB+536:D=BBB+487:G=40:BO=23:PUN=0
10 FE=10:LI=3
11 EL$="          PICCOLO             PICCOL
O    "
12 PRINT:PRINTTAB(14)"!←←←←← ←←←←←!"
13 PRINTTAB(14)"!↑↑↑££ £££££!"
14 PRINTTAB(14)"!←←←←← ←←←←←!"
15 PRINTTAB(14)"!↑££££ £↑↑↑↑!"
16 PRINTTAB(14)"!←←←←← ←←←←←!"
17 PRINTTAB(14)"!£££££ £££££!"
```

```
18 PRINTTAB(14)"!←←←←← ←←←←←!"
19 PRINTTAB(14)"!↑↑£££ ££££↑!"
20 PRINTTAB(14)"!←←←←← ←←←←←!"
21 PRINTTAB(14)"!↑↑↑↑£ £££↑↑!"
22 PRINTTAB(14)"!←←←←← ←←←←←!"
23 PRINT:PRINTTAB(14)"!←←←←← ←←←←←!"
25 PRINT:PRINT" !!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!"
26 PRINT" WIL JE INFORMATIE (J/N)*TTTT]"
27 PRINT"] "EL$
28 GET G$:IF G$="N"OR PEEK(56320)=111 TH
EN 100
29 IF G$="J" THEN 42
30 FOR O=0 TO 150:NEXT
31 EL$=RIGHT$(EL$,1)+LEFT$(EL$,LEN(EL$)-
1)
32 PL=BBB+100+M
33 POKE PL-40,32
34 POKE PL,35:POKE PL+R,0
35 POKE PL+40,34:POKE PL+R+40,2
36 POKE PL+80,32
37 M=M+G:IF M>400 OR M<=0 THEN G=-G
39 GOTO 27
42 PRINT"]J BENT PICCOLO IN EEN GROOT HO
TEL EN ";
43 PRINT"U  MOET SCHOENEN VERZAMELEN IN
DE           HOTELKAMERS"
44 PRINT"VOOR ELK PAAR ONTVANGT U 1 DOLL
AR"
45 PRINT"FOOI VAN DE GAST."
46 PRINT"ALS U MEER DAN 50 PAAR BIJ DE P
ORTIER"
47 PRINT"AFLEVERT, KRIJGT U EEN EXTRA DE
EL.";
49 PRINT"Q             [£] KAMER MET EEN PA
AR SCHOENEN"
50 PRINT"QQ   [↑] GESLOTEN DEUR"
51 PRINT"QQ   [ ] LEGE KAMER"
52 PRINT"QQ   [#] PICCOLO"
53 PRINT"QQ   [%] PORTIER"
54 PRINT"QQ   ["CHR$(34)"] LIFT"
56 IF PEEK(56320)<>111 THEN 56
100 BEELD$(1)=" !↑↑£££ ↑↑£££ ↑↑££↑ £££££
 £££££!"
110 BEELD$(2)=" !↑↑↑↑£ £££££ £↑£££ ££££↑
 £££££!"
```

```
120 BEELD$(3)=" !↑££££ ↑↑↑££ £££££ £££££
 £££↑!"
130 BEELD$(4)=" !£££££ £££££ £↑£££ ££↑↑£
 ££↑↑↑!"
140 BEELD$(5)=" !↑££££ £££££ £££££ £££££
 ↑↑↑↑↑!"
150 BEELD$(6)=" !£££££ ££↑↑↑ ££££↑ ££££↑
 £££↑!"
160 BEELD$(7)=" !£££££ £↑↑££ ££↑££ ££££↑
 £££↑↑!"
170 BEELD$(8)=" !£££££ ↑↑£££ £££££ ££↑££
 ££↑↑↑!"
180 BEELD$(9)=" !£££££ ↑↑£££ £££££ ↑↑↑↑£
 £££££!"
183 PRINT"♥";
184 PRINT" ■←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←
←←←"
185 PRINT" ■←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←
←←←"
187 PRINT" ▚!■←←←←← ←←←←← ←←←←← ←←←←← ←←
←←←▚!"
190 FORI=1TO9
195 W=INT(RND(1)*8)+1
200 PRINTBEELD$(W)
205 PRINT" ▚!■←←←←← ←←←←← ←←←←← ←←←←← ←←
←←←▚!"
210 NEXT
215 PRINT"▚ !                    %
   !"
217 PRINT"▚ !■←←←←← ←←←←← ←←←←← ←←←←← ←←
←←←▚!"
220 PRINT"■ ←←←←←←←←←←←←←←←←←←←←←←←←←←←←←←
←←←"
221 FOR I=BBB+77 TO BBB+917 STEP 40:POKE
 I,38:POKE I+R,6:NEXT
230 POKE X,35:POKE X+R,0:POKE X+1,32:POK
E X+2,32:POKE X-1,32:POKE X-2,32
250 POKE 54296,15
260 POKE 54277,31:POKE 54278,128:POKE 54
272,1:POKE 54273,1
270 PRINT"             HI-SCORE ";HIS;
280 PRINT"TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT"
1000 GOSUB 3000
1003 FOR I=1 TO FE:NEXT
1005 POKE D,32:POKE D+6,32:POKE D+12,32:
POKE D+18,32
1010 IF PEEK(D+G)=31 THEN G=-G
1020 D=D+G
```

```
1030 POKE D,34:POKE D+6,34:POKE D+12,34:
POKE D+18,34
1040 IF OL=0 THEN 2000
1050 POKE X,32:X=X+G
1060 IF PEEK(X)=31 THENX=X+40:GOTO 9000
1070 POKE X,35:POKE X+R,0
2000 BO=BO-0.05
2010 POKE INT(BO)*40+BBB+37,32:IF BO=<1
THEN 9000
2990 GOTO 1000
3000 IF PEEK(56320)=127 THEN H=0:RETURN
3010 IF PEEK(56320)=<111 THEN 3500
3020 IF PEEK(56320)=123 THENY=-1:GOTO 30
50
3030 IF PEEK(56320)=119 THEN Y=1:GOTO 30
50
3040 RETURN
3050 IF PEEK(X+Y)=31 THEN Y=Y-40:IF PEEK
(X+Y)=31 THEN 9000
3053 :IF PEEK(X+Y)=28 THEN POKE54276,129
:PUN=PUN+1:GOTO 3100
3054 IF PEEK(X+Y)=33 THEN RETURN
3055 IF PEEK(X+Y)=30 THEN RETURN
3057 IF PEEK(X+Y)=34 OR PEEK(X+Y+40)=34
THEN 3900
3060 IF PEEK(X+Y)=37 THEN 3800
3065 IF PEEK(X+Y)=32 AND PEEK(X+Y+40)=32
THEN:POKE X,32:X=X+Y:GOTO 9000
3070 IF PEEK(X+Y)=32 AND PEEK(X+Y+40)=10
2 THEN 3100
3100 POKE X,32
3110 X=X+Y:OL=0
3120 POKE X,35:POKE X+R,0
3130 PRINT"] ←PUNT←";PUN+TR;:PRINT"←←SCH
OENEN";PUN
3180 POKE 54276,0
3190 RETURN
3500 IF H=0 THEN G=-G:H=1
3510 RETURN
3800 FE=FE-1:IF PUN<50 THEN TR=TR+PUN:GO
TO 3810
3805 TR=TR+INT(PUN*(BO*7))
3810 X=BBB+536:D=BBB+487:G=40:BO=23:PUN=
0
3830 GOTO 183
3900 POKE X,32
3905 X=X+Y
3910 IF PEEK(X)=34 THEN X=X-40:IF PEEK(X
)=34 THEN 9000
```

```
3920 OL=1
3930 POKE X,35:POKE X+R,0
3940 RETURN
9000 TR=TR+PUN:LI=LI-1
9010 IF BO=<1 THEN GOTO 9020
9015 POKE X,32:X=X+40:IF PEEK(X+40)=31 T
HEN 9100
9017 POKE X,35:POKE X+R,0:GOTO 9015
9020 POKE X,32
9030 BO=23:FOR I=1 TO 23:POKE I*40+BBB+3
7,32:NEXT:OL=0
9035 D=BBB+487:G=40:BO=23:PUN=0
9036 IF LI=0 THEN 9800
9040 X=BBB+536:POKE X,35:POKE X+R,0:GOTO
 183
9100 POKE X,39:POKE X+R,0
9110 POKE 54277,31:POKE 54278,255:POKE 5
4272,0:POKE 54273,2:POKE 54276,33
9120 FOR I=1 TO 500:NEXT:POKE 54276,0
9190 GOTO 9020
9800 PRINT"J";:POKE 53281,1
9830 PRINT"QQQQQQQQQQQQQ              JE HEBT
 ";TR+PUN;"DOL.VERDIEND"
9840 IF TR+PUN>HIS THEN HIS=TR+PUN
9845 FOR X= 1 TO 1000
9850 TR=0:PUN=0
9880 GET A$:IF A$<>"" OR PEEK(56320)=111
THEN 6
9890 GOTO 9880
50000 DATA127,127,81,22,20,22,56,0
50001 DATA60,60,102,102,126,126,102,0:A
50002 DATA120,124,102,124,126,103,126,0:
B
50003 DATA60,126,102,96,102,126,60,0:C
50004 DATA124,126,99,99,99,126,124,0:D
50005 DATA127,126,96,124,96,126,127,0:E
50006 DATA127,126,124,96,124,120,112,0:F
50007 DATA62,121,112,115,115,63,31,0:G
50008 DATA115,115,115,127,127,115,115,0:
H
50009 DATA28,28,28,28,28,28,28,0:I
50010 DATA127,127,7,7,103,63,31,0:J
50011 DATA115,119,126,126,126,119,115,0:
K
50012 DATA112,112,112,112,112,126,127,0:
L
50013 DATA119,127,107,107,107,99,119,0:M
50014 DATA103,115,123,127,127,119,115,0:
N
```

```
50015 DATA62,127,119,119,119,127,62,0:O
50016 DATA126,127,115,127,126,112,112,0:
P
50017 DATA62,127,99,111,111,127,62,0:Q
50018 DATA126,127,115,127,124,126,119,0:
R
50019 DATA63,126,96,62,3,63,126,0:S
50020 DATA127,127,93,28,28,28,28,0:T
50021 DATA119,119,119,119,127,127,28,0:U
50022 DATA119,119,50,58,58,28,8,0:V
50023 DATA119,99,107,107,107,127,54,0:W
50024 DATA99,119,54,62,62,119,99,0:X
50025 DATA119,59,31,14,14,14,14,0:Y
50026 DATA127,63,7,28,112,126,127,0:Z
50027 DATA240,240,192,192,192,192,240,24
0:[
50028 DATA0,36,102,102,102,102,36,0:£
50029 DATA15,15,3,3,3,3,15,15:]
50030 DATA126,126,126,78,62,126,126,126:
↑
50031 DATA239,239,239,0,251,251,251,0:←
50032 DATA0,0,0,0,0,0,0,0
50033 DATA255,0,255,0,255,0,255,0:!
50034 DATA255,255,255,189,60,60,60,60:"
50035 DATA0,60,60,24,255,24,60,36:#
50036 DATA101,85,87,87,85,85,85,101:$
50037 DATA24,24,24,60,60,126,255,48:%
50038 DATA127,254,127,254,127,254,127,25
4:&
50039 DATA0,24,24,126,126,24,24,24:'
50040 DATA24,48,96,96,96,96,48,24:(
50041 DATA24,12,6,6,6,6,12,24:)
50042 DATA48,72,72,16,16,0,16,0:*
50043 DATA0,60,60,126,126,60,60,0:+
50044 DATA0,0,0,0,0,12,24,48:,
50045 DATA0,0,0,126,126,0,0,0:-
50046 DATA0,0,0,0,0,96,96,0:.
50047 DATA0,6,12,24,24,48,96,0:/
50048 DATA0,24,60,102,102,60,24,0:0
50049 DATA0,12,28,60,12,12,12,0:1
50050 DATA0,12,18,4,8,16,30,0:2
50051 DATA0,30,2,4,2,18,30,0:3
50052 DATA0,16,16,16,20,30,4,0:4
50053 DATA0,30,16,28,2,2,28,0:5
50054 DATA0,12,18,16,28,18,12,0:6
50055 DATA0,30,18,4,8,16,16,0:7
50056 DATA0,12,18,12,18,18,12,0:8
50057 DATA0,24,36,28,4,36,24,0:9
50058 DATA-1
```

7 INTERRUPTS

Interrupts zijn routines waar automatisch naartoe wordt gesprongen, als dat voor het goed functioneren van de computer nodoug is. Behalve de rasterinterrupt (TIPS & TRUCS 1) heeft de Commodore 64 in volgorde van prioriteit:

RESET
NMI
IRQ

De prioriteit is van belang als verschillende interruptroutines tegelijk worden aangeboden. Alle interrupts kunnen worden verhinderd.

7.1 RESET

SYS 64738 levert een RESET op. Adres 64738 staat als LSB en MSB in $FFFC-$FFFD. Maar het verdient de voorkeur de hardware te gebruiken, omdat een programma kan vastlopen. Invoer is dan niet meer mogelijk. Breng een knop op de computer aan, waarmee massa en de RESET-lijn van de processor met elkaar kunnen worden verbonden en voorzie de schakeling eventueel van een zoemer. Bij een RESET bepalen $FFFC-$FFFD, dat de computer naar $FCE2 springt. Vanaf dat adres loopt het programma als volgt:

```
FCE2 LDX #$FF
FCE4 SEI
FCE5 TXS
FCE6 CLD
FCE7 JSR $FD02
FCEA BNE $FCEF
FCEC JMP ($8000)
FCEF STX $D016
FCF2 JSR $FDA3
FCF5 JSR $FD50
FCF8 JSR $FD15
FCFB JSR $FFBD
FCFE CLI
FCFF JMP ($A000)
```

In adres $FCE4 wordt de interrupt disable flag gezet: de computer wordt niet meer gestoord. Een RESET kan niet worden verhinderd door RUN/STOP en RESTORE tegelijk in te drukken. In $FCE7 springt de computer naar een subroutine met beginadres $FD02:

```
FD02 LDX #$FF
FD04 LDA $FD0F,X
FD07 CMP $8003,X
FD0A BNE $FD0F
FD0C DEX
FD0D BNE $FD04
FD0F RTS
```

De computer leest een voor een bepaalde waarden uit het ROM en vergelijkt ze met de waarden in $8004-$8008. Zijn ze identiek, dan wordt een indirekte sprong uitgevoerd naar een adres waarvan het LSB in $8000 en het MSB in $8001 staat. Bevat $8000 bijvoorbeeld $00 en $8001 $60, dan gaat het programma verder bij $6000. Wilt u dus een programma bij RESET laten beginnen, dan moeten $8000-$8008 er als volgt uitzien:

```
8000 LB Sprongvector bij reset
8001 HB
8002 LB Sprongvector bij nmi(zie nmi)
8003 HB
8004 195 "C" :bit 7 gezet
8005 194 "B" :bit 7 gezet
8006 205 "M" :bit 7 gezet
8007  56 "8"
8008  48 "0"
```

Moet de computer direkt na het inschakelen een programma uitvoeren, adresseer dan een EPROM in dit bereik en stel de waarden in zoals hierboven. Volgens dit principe werken ook de modules, die zich bij het inschakelen meteen melden. Als de computer bij $8000 geen programma aantreft, gaat hij verder bij $FCEF. In de subroutines daarna worden de pages 0, 2, 3, het I/O-bereik en het videobereik opnieuw geïnitialiseerd. Tenslotte volgt een sprong naar de koude start van BASIC. Als u met RESET een BASIC-programma wilt starten, moeten $8000-$8001 op de volgende instructies wijzen:

```
JSR $A659
JMP $A7AE
```

Ze zijn het machinetaalequivalent van de BASIC-instructie RUN.

7.2 NMI

$0318-$0319 (792-793) bepalen het sprongadres van NMI, meestal $FE47 (65095). De NMI-routine begint bij $FE43 (65091):

```
FE43 SEI
FE44 JMP ($0318)
FE47 PHA
FE48 TXA
FE49 PHA
FE4A TYA
FE4B PHA
FE4C LDA #$7F
FE4E STA $DD0D
FE51 LDY $DD0D
FE54 BMI $FE72
FE56 JSR $FD02
FE59 BNE $FE5E
FE5B JMP ($8002)
FE5E JSR $F6BC
FE61 JSR $FFE1
FE64 BNE $FE72
FE66 JSR $FD15
FE69 JSR $FDA3
FE6C JSR $E518
FE6F JMP ($A002)
```

In $FE43 wordt de interrupt disenable flag gezet. Wilt u vector $0318-$0319 op een ander programma laten wijzen, dan moet de instructie CLI in dat programma staan. Anders is de IRQ verhinderd (zie paragraaf 7.3). De NMI-vector wijst op RESET door:

```
POKE 792,226: POKE 793,252
```

Druk op RESTORE (=NMI) en de RESET is een feit. Daarna nog een keer RESTORE zet vector $0318-$0319 weer in zijn oude stand: het effect is ongedaan gemaakt. $FE47-$FE4B bewaren de inhoud van de accumulator, het X- en Y-register. In $FE54 (65108) wordt gecontroleerd of de RS-232 interface actief is. Als dat zo is, springt de computer naar $FE27 (65138), het beginadres van de NMI-routine voor de RS-232. In $FE56 (65110) wordt nagegaan of er een moduul staat vanaf $8000. Zo ja, dan staat in $8002 het LSB en in $8003 het MSB van het adres, waar met JMP($8002) naartoe moet worden gesprongen. Omdat RESET en NMI op de waarden in $8000 reageren, kunt u twee programma's starten. Het een met RESET, het andere met RESTORE. In $FE61 (65121) wordt de STOP-toets

ingelezen. Als die is ingedrukt, gaat de computer naar $FE66 (65126), initialiseert het I/O-bereik opnieuw, wist het scherm en springt naar de Basic warme start. Meestal heeft RESTORE geen effect. De STOP-toets wordt altijd eerst ingelezen. Is die niet ingedrukt, dan wordt er naar de NMI-routine voor de RS-232 gesprongen. Breng deze kennis in de praktijk door de volgende regels in te lezen:

```
10 FOR X=0 TO 8: READ A: POKE 32768+X,A: NEXT X
20 20 DATA 68,229,53,164,195,205,56,48

RUN
```

De adressen $8000-$8008 zien er zo uit:

32768	68	32772	195
32769	229	32773	194
32770	53	32774	205
32771	164	32775	56
		32776	48

In 32768-32769 staat het adres (58692), waar de computer naartoe springt bij een RESET. Daar begint een subroutine, die het beeldscherm wist. Bij een NMI springt de computer naar 42037 (in 32770-32771), waar een subroutine begint, die OUT OF MEMORY uitvoert. In 32772-32776 staan de codes, die de computer nodig heeft om te herkennen dat er vanaf $8000 een ROM aanwezig is. Als u, nadat de twee regels zijn gestart, op RESTORE drukt, verschijnt de foutmelding OUT OF MEMORY. Dat de cursor soms op zijn plaats blijft komt, omdat de IRQ-routine de cursor beheert. De NMI-routine heeft een hogere prioriteit en wordt dus aangesproken voordat de IRQ-routine is afgelopen. Omdat RESET het scherm wist, is het een goede gelegenheid om een programma te starten. U kunt bijvoorbeeld een programma zodanig opslaan vanaf $8000, dat het na het laden met RESET of RESTORE wordt gestart. Vreemde gebruikers kunnen het programma moeilijk starten, want er is geen BASIC-regel, die het begin aangeeft. De beveiliging wordt nog beter, als er na RESET of RESTORE een codewoord moet worden gegeven, zonder dat de computer daarnaar vraagt. Ook kunt u nog een autostart inbouwen.

7.3 IRQ

Het beginadres van de IRQ-routine in $0314-$0315 (788-789) kan worden veranderd:

```
POKE 788,226: POKE 789,252
```

Druk op RETURN en ...226+252*256=64738... de computer voert een RESET (adres 64738) uit. De IRQ-routine wordt elke 1/60 seconde aangesproken en niet pas nadat een schakelaar contact heeft gemaakt, zoals bij RESET en NMI. IRQ doet alles wat steeds weer gebeuren moet: de tijd bijhouden, de cursor laten knipperen, de STOP-toets inlezen enzovoort.

```
EA31  JSR $FFEA
EA34  LDA $CC
EA36  BNE $EA61
EA38  DEC $CD
EA3A  BNE $EA61
EA3C  LDA #$14
EA3E  STA $CD
EA40  LDY $D3
EA42  LSR $CF
EA44  LDX $0287
EA47  LDA ($D1),Y
EA49  BCS $EA5C
EA4B  INC $CF
EA4D  STA $CE
EA4F  JSR $EA24
EA52  LDA ($F3),Y
EA54  STA $0287
EA57  LDX $0286
EA5A  LDA $CE
EA5C  EOR #$80
EA5E  JSR $EA1C
EA61  LDA $01
EA63  AND #$10
EA65  BEQ $EA71
EA67  LDY #$00
EA69  STY $C0
EA6B  LDA $01
EA6D  ORA #$20
EA6F  BNE $EA79
EA71  LDA $C0
EA73  BNE $EA7B
EA75  LDA $01
```

```
EA77  AND #$1F
EA79  STA $01
EA7B  JSR $EA87
EA7E  LDA $DC0D
EA81  PLA
EA82  TAY
EA83  PLA
EA84  TAX
EA85  PLA
EA86  RTI
```

In $EA31 (59953) wordt naar een subroutine (beginadres F69B (63131)) gesprongen, die de tijd bijhoudt en de STOP-toets inleest. Als u deze subroutine buiten beschouwing laat, wordt de tijd niet meer bijgehouden en functioneert RUN/STOP alleen nog als de cassetterecorder wordt gebruikt. In dat geval wordt RUN/STOP niet via IRQ ingelezen.

```
POKE 788,52
```

laat de IRQ-routine beginnen bij $EA34 (59956). Lees dan het volgende programmaatje in:

```
10 PRINT TI$
20 GOTO 10
```

Na het starten blijft TI$ staan op het tijdstip, waarop de POKE-instructie werd ingevoerd. Ook STOP functioneert niet meer, maar het tegelijk indrukken van RUN/STOP en RESTORE heeft wel effect. NMI wordt aangesproken met RESTORE. Er wordt gecontroleerd of STOP is ingedrukt. De IRQ-routine is verhinderd. In $EA36 wordt nagegaan of de cursor paraat is. Is dat niet zo, dan springt de computer naar $EA61 (60001). Als de cursor wel paraat is, gaat het programma gewoon verder. $CC (204) bepaalt of de cursor aan of uit is.

```
10 POKE 204,0 : REM CURSOR AAN
20 GET A$: IF A$="" THEN 20
30 POKE 204,1 : REM CURSOR UIT
40 WAIT 198,1
```

Als u dit programma probeert, verschijnt er een cursor aan het begin van de regel na RUN. Zo kan de gebruiker op een GET-instructie gewezen worden. Een nadeel is, dat de cursor bevriest als u regel 20 verlaat (op een toets drukt), wanneer de cursor net op het scherm is verschenen. Een nieuwe cursor verschijnt pas als op een volgende toets is gedrukt. Dat kan worden verholpen met

```
IF PEEK(207)=1 THEN 25
```

Geheugenplaats 207 geeft aan of de cursor op het scherm is (=1) of niet (=2). In $EA38 (60001) wordt de timer voor de cursor met 1 verminderd. Is die niet gelijk aan 0, dan springt de computer naar $EA61 (60001). Bij 0 wordt hij op $14 (20) gezet. De computer moet IRQ 20x aanspreken om de cursor te veranderen. De cursor knippert daarom in intervallen van 20x1/60=1/3 seconde. Dat kan worden gewijzigd:

```
JSR $FFEA ; TIJD, STOP-TOETS
LDA $CC   ; CURSOR AAN?
BNE *     ; NEE
DEC $CD   ; TIMER VERLAGEN
BNE *     ; ONGELIJK NUL, VERDER
LDA $FF   ; TIMER OPNIEUW INSTELLEN
JMP $EA3E ; IRQ VERDER
* JMP $EA61    ; NORMAAL VERDER
```

In BASIC:

```
10 FOR X=0 TO 18: READ A: POKE 24625+X,A: NEXT X
20 DATA  32,234,255,165,204,208,  9,198,205,208
30 DATA   5,165,255, 76, 62,234, 76, 97,234
```

Voer na RUN

```
POKE 788,96
```

in om de IRQ-routine naar het nieuwe programma te laten wijzen. Een 0 in geheugenplaats 255 levert een bevroren cursor op. Bij 1 beweegt de cursor heel snel, bij 255 heel langzaam. Omdat adres 255 doorlopend wordt gelezen, beïnvloedt de invoer van een andere waarde meteen het knipperen van de cursor. In de adressen tot $EA5E (59998) werkt de computer de cursorfuncties af, dat wil zeggen dat hij de cursor uitvoert, het teken dat daaronder staat onthoudt evenals de kleur, enzovoort.

```
POKE 788,97
```

laat het gedeelte van de IRQ-routine dat de cursor betreft, buiten beschouwing: de tijd wordt niet meer bijgehouden, de STOP-toets niet ingelezen en de cursor niet uitgevoerd. Er blijven tekens verschijnen, omdat IRQ het toetsenbord pas later uitleest. $EA61-$EA79 (60001-60025) zijn voor de recorder. Als een toets is ingedrukt, wordt de motor aangezet. Is dat niet het geval, dan is de motor uit:

1. Lees POKE 788,123 (begin van IRQ=$EA7B) in. Als u op PLAY drukt, begint de motor niet te lopen.

2. Druk tegelijk op RUN/STOP en RESTORE om weer in de standaardmodus te komen. Druk vervolgens op PLAY en lees POKE 788,123 in. De STOP-toets van de recorder heeft geen effect, de motor blijft lopen.

In $EA7B (60027) wordt naar een subroutine gesprongen, die het uitlezen van het toetsenbord voor zijn rekening neemt en een teken inleest. Als u dit adres ook nog buiten beschouwing laat,

```
POKE 788,126
```

is de computer volledig hulpeloos. RUN/STOP-RESTORE functioneert nog, omdat RESTORE direkt wordt ingelezen, in tegenstelling tot de andere toetsen. RESTORE is bijna onmiddellijk verbonden met pin 4 (NMI) van de CPU. In $EA81-$EA85 (60033-60037) worden de oorspronkelijke waarden van de registers weer ingesteld. RTI (return from interrupt) sluit de IRQ-routine af. IRQ kan gemakkelijk naar allerlei routines wijzen en vervult daarom veel functies. Vergeet nooit aan het einde van een eigen routine naar de oorspronkelijke IRQ terug te keren, anders zijn STOP, klok, cursor, recorder en toetsenbord buiten werking.

7.4 Het programmeren van de IRQ

Voor het programmeren van een andere interruptroutine moet de vector van 788-789 worden veranderd. IRQ kan elke 1/60 seconde vanuit de direkte modus en de programmamodus worden aangesproken. Het volgende programma laat zien welke mogelijkheden de programmering van IRQ biedt en hoe snel deze routine werkt.

```
10 DATA 120,169, 15,141, 20,  3,169
20 DATA 159,141, 21,  3,133, 56, 88
30 DATA  96,238, 32,208, 76, 49,234
40 FOR I= 40704 TO 40724
50 READ A
60 POKE I,A
70 S=S+A
80 NEXT I
90 IF S <> 2171 THEN PRINT"FOUT IN DATA!":END
100 PRINT"DATA OK": SYS40704
110 NEW
```

Als de interrupt wordt aangesproken, verhoogt dit programma de kleur van het beeldschermkader met 1. Dat gebeurt zo vaak, dat er van een bepaalde kleur geen sprake is. Een goed geprogrammeerde interrupt heeft geen invloed op de snelheid waarmee wordt gewerkt. Nog steeds moeten er BASIC-regels kunnen worden geschreven en bewerkt. Als het u stoort dat het beeldschermkader knippert, druk dan op RUN/STOP-RESTORE. Start het programma met

```
SYS 40704
```

De routine bestaat uit twee delen. In het eerste deel wordt de IRQ-vector veranderd en er wordt verhinderd dat over de routine kan worden heen geschreven. In BASIC kunt u de vectoren niet zomaar wijzigen. Als een programma bij 40719 (hier: begin van het tweede deel) begint en u leest

```
POKE 788,15: POKE 789,159
```

in, dan springt de computer, nadat 788 is veranderd, naar $EAOF en niet naar $EA31. Meestal loopt het programma vervolgens vast. Gewoonlijk veranderen interruptroutines in machinetaal de vectoren automatisch in het eerste deel. De machinetaalinstructie SEI (set interrupt) verhindert, dat een interrupt wordt uitgevoerd. Na CLI (clear interrupt) voert de computer de interrupt uit met nieuwe waarden. Bovendien wordt in het eerste deel het eind van het BASIC-geheugen aan het begin van het machinetaalprogramma gezet, zodat de routine niet kan worden gewist door variabelen of door het BASIC-programma. De machinetaalinstructies uit het eerste deel:

```
SEI             ; VERHINDERT INTERRUPT
LDA #           ; HIER WORDT DE LOW-BYTE VAN
                  HET BEGINADRES VAN DEEL 2 BEPAALD
STA $ 0314      ; SLAAT LOW-BYTE OP IN IRQ-WIJZER
LDA #           ; BEPAALD HIGH-BYTE VAN DEEL 2
STA $ 0315      ; SLAAT HIGH-BYTE OP IN IRQ-WIJZER
STA $ 38        ; ZET HIGH-BYTE GEHEUGENEINDE OP
                ; BEGIN ROUTINE, OP DEZE WIJZE
                ; WORDT DE ROUTINE AFGEBAKEND
CLI             ; INTERRUPT WORDT WEER UITGEVOERD
```

Net als bij de pieptoonroutine in de volgende paragraaf kunnen in het eerste deel waarden worden ingesteld die de basis vormen voor het tweede deel. Het tweede deel is de eigenlijke interruptroutine. De computer spreekt alleen dit gedeelte aan. Zorg ervoor dat het niet te lang wordt, anders vermindert de snelheid van de direkte modus. De sprong naar $EA31 aan het eind van de routine voorkomt, dat het programma vastloopt.

7.5 Toetsenbordpiep

Bij de invoer van programma's en tabellen is een piepje erg handig. Met een IRQ-routine kan deze faciliteit op de Commodore 64 worden aangebracht:

```
  0 REM PIEPTOON-ROUTINE
 10 FOR I=0 TO 61
 20 READ A
 30 S=S+A
 40 POKE 40704+I,A
 50 NEXT I
 60 IF S <>6973 THEN PRINT"FOUT IN DATA!":END
 70 PRINT"DATA OK"
 80 SYS 40704
 90 DATA 169,255,141,  6,212,141, 24,212,169
100 DATA   9,141,  5,212,169,103,141,  1,212
110 DATA 169, 33,141,  0,212,120,169, 38,141
120 DATA  20,  3,169,159,141, 21,  3,133, 56, 88
130 DATA  96, 72,165,203,201, 64,208,  9,169
140 DATA   0,141,  4,212,104, 76, 49,234,169
150 DATA  17,141,  4,212, 76, 50,159
```

De listing in assembler:

```
9F00  LDA #$FF
9F02  STA $D406
9F05  STA $D418
9F08  LDA #$09
9F0A  STA $D405
9F0D  LDA #$67
9F0F  STA $D401
9F12  LDA #$21
9F14  STA $D400
9F17  SEI
9F18  LDA #$26
9F1A  STA $0314
9F1D  LDA #$9F
9F1F  STA $0315
9F22  STA $38
9F24  CLI
9F25  RTS
9F26  PHA
9F27  LDA $CB
9F29  CMP #$40
9F2B  BNE $9F36
```

```
9F2D  LDA #$00
9F2F  STA $D404
9F32  PLA
9F33  JMP $EA31
9F36  LDA #$11
9F38  STA $D404
9F3B  JMP $9F32
```

In het deel $9F17-$9F25 worden net als in de vorige paragraaf de vectoren veranderd. Behalve de golfvorm worden de waarden voor de geluidsproduktie ingesteld in la ($9F00-$9F16). Ze spelen geen rol in de eigenlijke interruptroutine. In adres 203 (deel 2) wordt nagegaan of een toets is ingedrukt. Is dat niet het geval (waarde=64), dan krijgt de golfvorm de waarde 0: er is geen toon te horen. Is er wel op een toets gedrukt (waarde ongelijk 64), dan wordt de golfvorm 17 en hoort u een pieptoon. Daarna springt het programma naar $EA31 (59953).

7.6 Muziek

Een muziekprogramma ziet er al gauw omvangrijk uit. Dat komt door de programmering van de noten. Elke noot in het volgende programma neemt 3 waarden in beslag: frequentie low, frequentie high en de duur van de noot. In het eerste deel wordt het programma geladen, in het tweede deel de duur van de noten ingelezen. Sla het programma op, voordat u het probeert.

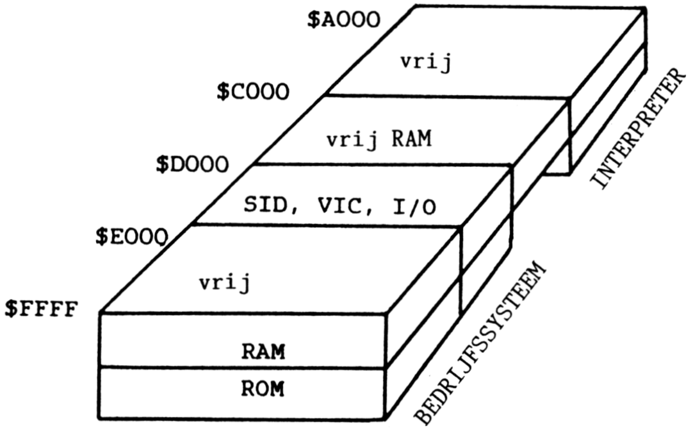
```
  0 REM MUZIEK MET IRQ
 10 DATA 120,169, 38,141, 20,  3,169,144
 20 DATA 141, 21,  3, 88,133, 56,169,  0
 30 DATA 141,243,159,169,  1,141,241,159
 40 DATA 169,  0,141, 20,212,169, 31,141
 50 DATA  24,212,141, 19,212, 96,206,241
 60 DATA 159,208, 48, 72,138, 72,169,  0
 70 DATA 141, 18,212,174,243,159,189,  0
 80 DATA 145,141, 14,212,189,  0,146,141
 90 DATA  15,212,189,  0,147,141,241,159
100 DATA 169, 33,141, 18,212,232, 56,224
110 DATA  60,144,  2,162,  0,142,243,159
120 DATA 104,170,104, 76, 49,234
130 FOR I=36864 TO 36957
140 READ A
150 POKE I,A
```

```
160 S=S+A
170 NEXT
180 IF S<>11531 THEN PRINT"FOUT IN DATA":END
190 INPUT"HOEVEEL NOTEN";N:POKE36944,N
200 REM NOTEN : ^=60 NOTEN !!
210 FOR I= 0 TO 59
220 READ L
230 POKE 37120+I,L
240 READ H
250 POKE 37376+I,H
260 READ W
270 POKE37632+I,W
280 NEXT
290 SYS 36864
300 REM MUZIEK-GEGEVENS
310 DATA  196,  9, 10, 10, 13, 10, 10, 13, 10
320 DATA  162, 14, 10,109, 16, 10, 10, 13, 10
330 DATA  109, 16, 10,162, 14, 10,196,  9, 10
340 DATA   10, 13, 10, 10, 13, 10,162, 14, 10
350 DATA  109, 16, 10, 10, 13, 20,158, 11, 10
360 DATA  196,  9, 10, 10, 13, 10, 10, 13, 10
370 DATA  162, 14, 10,109, 16, 10,103, 17, 10
380 DATA  109, 16, 10,162, 14, 10, 10, 13, 10
390 DATA  158, 11, 10,196,  9, 10,247, 10, 10
400 DATA  158, 11, 10, 10, 13, 20, 10, 13, 10
410 DATA    0,  0, 10,247, 10, 15,158, 11,  5
420 DATA  247, 10, 10,196,  9, 10,247, 10, 10
430 DATA  158, 11, 10, 10, 13, 20,196,  9, 15
440 DATA  247, 10,  5,196,  9, 10,180,  8, 10
450 DATA   55,  8, 10,180,  8, 10,196,  9, 20
460 DATA  247, 10, 15,158, 11,  5,247, 10, 10
470 DATA  196,  9, 10,247, 10, 10,158, 11, 10
480 DATA   10, 13, 10,247, 10, 10,196,  9, 10
490 DATA   10, 13, 10,158, 11, 10,162, 14, 10
500 DATA   10, 13, 20, 10, 13, 10,  0,  0, 99
```

Na de start van het programma vraagt de computer hoeveel noten het melodietje heeft. In dit geval zijn dat er 59. Vervolgens worden ze ingelezen en wordt de routine gestart. U hoort YANKEE DOODLE. Bij het programmeren speelt de muziek gewoon door. De melodie loopt via stem 3 en kan dus bij een spelletje als speciaal effect dienen. De noten (maximaal 256) in de DATA-regels hebben de vorm: frequentie low, frequentie high en duur. In YANKEE DOODLE is de kwart noot 10, de halve 20 en de hele 40.

8 BEDRIJFSSYSTEEM: ROM IN RAM

Het bedrijfssysteem van de Commodore 64 kan in het RAM met hetzelfde adresbereik worden gezet.



In bepaalde geheugenbereiken zit een RAM- en een ROM-geheugen. $A000 is dus een RAM- en een ROM-adres. De computer kan er maar een tegelijk adresseren. In dit hoofdstuk behandelen we $A000-$BFFF (BASIC-interpreter) en $E000-$BFFF (I/O-bereik).

8.1 Kopieerroutines

Het ROM bevat interpreter en I/O-bereik, het RAM in deze bereiken is leeg. Het BASIC-geheugen wordt gewist als de computer uit wordt gezet. het ROM kan in het RAM worden gekopieerd met PEEKs en POKEs. Een kopieerroutine vcoor de interpreter:

```
10 FOR A=10*4096 TO 12*4096 -1
20 POKE A,PEEK(A)
30 NEXT A
40 END
```

En voor het bedrijfssysteem:

```
10 FOR A=14*4096 TO 16*4096 -1
20 POKE A,PEEK(A)
30 NEXT A
40 END
```

Voorkom, dat de computer vastloopt: schakel het geheugenbereik pas naar het RAM, als het ROM in het RAM is gekopieerd.

INT.ROM	INT.RAM	BEDR.ROM	BEDR.RAM	omschakeling
1	0	1	0	POKE 1,55
0	1	1	0	POKE 1,54
0	1	0	1	POKE 1,53
1	0	0	1	....

In BASIC kunt u niet het bedrijfssysteem alleen in het ROM laten werken. Het duurt lang voordat de BASIC-routine klaar is met kopiëren. Een machinetaalprogramma doet dat veel sneller:

```
033C LDX#$20
033E LDA#$A0
0340 LDY#$00
0342 STY$22
0344 STA$23
0346 LDA($22),Y
0348 STA($22),Y
034A INY
034B BNE$0346
034D INC$23
034F DEX
0350 BNE$0346
0352 RTS
```

In BASIC:

```
10 FOR I=828 TO 851
20 READ X: POKE I,X: NEXT
30 DATA 162, 32,169,160,160,  0,132, 34,133, 35,177, 34
40 DATA 145, 34,200,208,249,230, 35,202,208,244, 96
```

```
50 SYS 828
```

U roept de routine op met SYS 828. Moeten de interpreter en het I/O-bereik met dit programma worden gekopieerd, verander dan regel 50:

```
50 SYS 828: POKE 831,224: SYS 828
```

In het boek 64 INTERN uit de reeks DATA BECKER/NEDERLANDS* staat de complete assemblerlisting van interpreter en I/O-bereik. Als ze in het RAM staan, kunnen die bereiken worden gewijzigd.

9 ROUTINES VAN HET BEDRIJFSSYSTEEM

Assemblerinstructies zijn niet zo veelzijdig als BASIC-instructies. PRINT in machinetaal vertalen is al een behoorlijk probleem. Maar een hoop dingen zijn doodeenvoudig, omdat veel routines in het bedrijfssyteem kant en klaar voorhanden zijn. Het bedrijfssysteem bestaat uit de interpreter en het I/O-bereik. De interpreter vertaalt BASIC-instructies in voor de computer begrijpelijke machinetaal. Het I/O-bereik verzorgt alle operaties met de buitenwereld.

Blokverschuivingsroutine, sprongadres: $A3B8 (41912)

Deze routine verplaatst een geheugenbereik snel naar andere adressen. Het begin van het oude blok moet in $5F-$60 (95-96) staan, het einde (+1) in $5A-$60 (90-91) en het einde van het nieuwe blok (+1) in $58-$59 (88-89).

Genoeg plaats in stack? sprongadres: $A3FB (41979)

Deze routine gaat na of er voldoende ruimte is in de stack. Is dat niet het geval, dan volgt de foutmelding OUT OF MEMORY.

Uitvoer van OUT OF MEMORY, sprongadres $A435 (42037)

Na de uitvoer komt de computer terug in de direkte modus (ready mode).

Foutmelding uitvoeren, sprongadres: $A43A (42042)

Na de uitvoer komt de computer weer terug in de direkte modus. Het nummer van de fout moet aan het X-register worden doorgegeven:

NUMMER dec.	hex.	FOUTMELDING
1	1	TOO MANY FILES
2	2	FILE OPEN
3	3	FILE NOT OPEN
4	4	FILE NOT FOUND
5	5	DEVICE NOT PRESENT
6	6	NOT INPUT FILE
7	7	NOT OUTPUT FILE
8	8	MISSING FILE NAME
9	9	ILLEGAL DEVICE NUMBER
10	A	NEXT WITHOUT FOR
11	B	SYNTAX
12	C	RETURN WITHOUT GOSUB
13	D	OUT OF DATA
14	E	ILLEGAL QUANTITY
15	F	OVERFLOW
16	10	OUT OF MEMORY
17	11	UNDEF'D STATEMENT
18	12	BAD SUBSCRIPT
19	13	REDIM'D ARRAY
20	14	DIVISION BY ZERO
21	15	ILLEGAL DIRECT
22	16	TYPE MISMATCH
23	17	STRING TOO LONG
24	18	FILE DATA
25	19	FORMULA TOO COMPLEX
26	1A	CAN'T CONTINUE
27	1B	UNDEF'D FUNCTION
28	1C	VERIFY
29	1D	LOAD
30	1E	BREAK

Deze routine bestaat ook in BASIC:

```
POKE 781,21: SYS 42042
```

betekent bijvoorbeeld ILLEGAL QUANTITY ERROR.

BASIC-regels opnieuw koppelen, sprongadres: $A533 (42291)

Regels uit een BASIC-programma worden in de juiste volgorde gezet.

NEW, sprongadres: $A642 (42562)

De BASIC-instructie NEW. Een BASIC-programma wordt daardoor verborgen.

CLR, sprongadres: $A65E (42590)

De BASIC-instructie CLR. Alle variabelen worden gewist.

RUN

Voor RUN is er geen routine in het bedrijfssysteem. De BASIC-instructie RUN in machinetaal:

```
JSR $A659
JSR $A7AE
```

STOP-toets, sprongadres: $A82C (43052)

Is er op de STOP-toets gedrukt, dan wordt een lopend BASIC-programma afgebroken. Een STOP-instructie in een BASIC-routine leidt tot een sprong naar adres $A8F2 (43055). Op die manier kan dus in machinetaal een BASIC-programma worden onderbroken.

GOTO, sprongadres: $A8A3 (43174)

De GOTO-routine begint eigenlijk op $A8A0 (43171). Dit adres bevat een sprong naar een subroutine, die het geldige regelnummer in de adressen $14-$15 haalt. U kunt het regelnummer zelf bepalen door naar $A8A3 te springen en het MSB en LSB in S14 en $15 te zetten.

String uitvoeren, sprongadres: $AB1E (43806)

Het adres van de 0, die het einde van de string aangeeft, moet in het Y-register (LSB) en de accumulator (MSB) komen te staan. Het volgende programma voert de cassettebuffer uit tot aan de eerste 0.

```
LDA #$41
LDY #$03
JSR $AB1E
RTS
```

De naam van een programma komt u zo gemakkelijk te weten. In BASIC doet u dat als volgt:

```
POKE 780,65: POKE 782,3: SYS 43806
```

Enkele strings in het ROM:

melding	accu (LSB)	Y-reg. (MSB)	adres
O.K.	100	163	$A364
ERROR (zonder CR)	105	163	$A368
IN	113	163	$A370
READY	118	163	$A378
BREAK (zonder CR)	129	163	$A380
BASIC BYTES FREE	96	228	$E467
(inschakelmelding)	115	228	$E473

Uitvoer van een spatie/cursor naar rechts, sprongadres: $AB3B (43835)

In het geval van een file (adres $13 (19) ongelijk 0) wordt een spatie uitgevoerd. In dit subprogramma zit nog een andere routine verborgen. Op het adres $AB45 staat de instructie voor het invoeren van de waarde van de volgende byte ($3F (63): vraagteken) in de accu, waarna op adres $AB47 de spronginstructie naar $E10C volgt voor het uitvoeren van dit vraagteken.

Haakje na teken gesloten? sprongadres: $AEF7 (44791)

Als het teken, waarop vector $7A-$7B (122-123) wijst, niet met een haakje is afgesloten, verschijnt SYNTAX ERROR op het scherm. Vervolgens gaat de computer in de direkte modus.

De andere routines in dit programma zijn:

(1) $AEFA (44794) als $AEF7, maar dan: haakje open?
(2) $AEFD (44797) als $AEF7, maar dan: komma?
(3) $AEFF (44799) als $AEF7, maar dan: teken in accu?

SYNTAX ERROR, sprongadres: $AF08 (44808)

Na SYNTAX ERROR gaat de computer in de direkte modus.

Is teken letter? sprongadres $B113 (45331)

Is het teken in de accu een letter, dan wordt de carry-flag gewist.

FAC-getal in integer getal veranderen, sprongadres: ($B1AA (45482)

Het LSB en MSB van het integere getal worden opgeslagen in $64-$65 (100-101).

BAD SUBSCRIPT ERROR, sprongadres $B245 (45637)

Na een BAD SUBSCRIPT ERROR gaat de computer in de direkte modus.

ILLEGAL QUANTITY ERROR, sprongadres $B248 (45640)

Na een ILLEGAL QUANTITY ERROR gaat de computer in de direkte modus.

Direkte modus? sprongadres $B3A6 (45990)

De direkte modus leidt tot ILLEGAL DIRECT ERROR. In deze routine zitten nog twee andere:

(1) $B3AB (45995) ILLEGAL DIRECT ERROR
(2) $B3AE (45998) UNDEF/D FUNCTION ERROR

FORMULA TOO COMPLEX ERROR, sprongadres: $B4D0 (46288)

FAC veranderen in LSB/MSB, sprongadres $B7F7 (47095)

Het getal in de FAC moet positief zijn en mag niet groter zijn dan 65536. Het LSB en MSB van zijn 16-bits equivalent komt in $14-$15 (20-21) en het Y-register te staan.

OVERFLOW ERROR, sprongadres $B97E (47486)

DIVISION BY ZERO ERROR, sprongadres $BB8A (48010)

LSB/MSB veranderen in integer getal, sprongadres $BDCD (48589)

De lobyte moet in de accu staan, de hibyte in het X-register. In BASIC:

```
10 INPUT"LOW-BYTE";LO
20 INPUT"HIGH-BYTE";HI
30 POKE 781,LO:POKE780,HI:SYS48589
40 PRINT:GOTO 10
```

In deze routine zitten er nog twee verborgen:

(1) $BDD4 Het integere getal wordt veranderd in een reëel getal
(2) $BDD7 Het reële getal wordt veranderd in een ASCII-string

BREAK ERROR, sprongadres: $E107 (57607)

Andere tekens? sprongadres: $E211 (57873)

Geen andere tekens resulteert in SYNTAX ERROR. Wilt u voordien controleren of er een komma is, gebruik dan $E2E0 (57870) als sprongadres.

BASIC koude start, sprongadres: $E394 (58260)

Deze routine is een gedeeltelijke reset:

(1) BASIC-programma's en variabelen worden gewist
(2) Het RAM wordt in de uitgangspositie gebracht,
(3) evenals $0300-$030B (768-779)
(4) Het beginbeeld verschijnt
(5) Sprong naar BASIC warme start

Kleuren en $0314-$0333 (788-819), waarin onder andere de sprongvectoren voor IRQ en NMI liggen, veranderen bijvoorbeeld niet.

Wachten op Commodore-toets, sprongadres: $E4E0 (58592)

Als deze toets na een bepaalde tijd niet wordt ingedrukt, springt de computer vanzelf weer terug.

Beeldscherm-reset, sprongadres: $E518 (58648)

De oorspronkeleijke kleuren worden ingesteld, het scherm wordt gewist en de cursor komt weer in de uitgangspositie. De routine kan worden onderverdeeld in:

(1) Beeldscherm-reset zonder dat de videocontroller wordt veranderd, sprongadres: $E51B (58651)
(2) Beeldscherm wissen, sprongadres: $E544 (58692)
(3) Cursor home, sprongadres: $E566 (58726)
(4) Cursor home en videocontroller opnieuw initialiseren, sprongadres: $E59A (58778)
(5) Alleen videocontroller initialiseren, sprongadres: $E5A0 (58784)

Terug naar vorige regel, sprongadres: $E701 (59137)

Beeldscherm scrollen, sprongadres $E8EA (59626)

Het scherm verschuift een regel naar boven. De bovenste regel verdwijnt en er komt een nieuwe bij.

Beeldschermregel wissen I, sprongadres ($E9FF)

Het regelnummer moet in het X-register worden overgedragen. De bovenste regel is regel 0. In BASIC:

```
POKE 781, (regelnummer): SYS 59903
```

Beeldschermregel wissen II, sprongadres $E9FF (59905)

Net als boven wordt de regel gewist, waarvan het nummer in het X-register staat. Als de regel maar tot een bepaalde kolom moet worden gewist, moet het kolomnummer (0-39) in het Y-register komen te staan:

```
LDA #$0A
LDX #$00
JSR $E9FF
RTS
```

Dit machinetaalprogramma en zijn BASIC-equivalent wissen de eerste regel (nummer 0) tot aan de elfde positie (nummer 10).

Vertraging van 1 milliseconde, sprongadres $EEB3 (60958)

Meldingen van het bedrijfssysteem uitvoeren, sprongadres $F12B (61739)

De strings, die bij gebruik van diskdrive en datasette een rol spelen, worden uitgevoerd. Eerst wordt gecontroleerd of de computer zich in de programmamodus bevindt. Is dat het geval, dan wordt er niets uitgevoerd. De offset van de melding moet in het Y-register komen te staan.

Y-REGISTER dec.	hex.	MELDING
0- 1	00-01	I/O ERROR #
12- 13	0C-0D	SEARCHING
23	17	FOR
27- 28	1B-1C	PRESS PLAY ON TAPE
46	2E	PRESS RECORD & PLAY ON TAPE
73- 74	49-4A	LOADING
81- 82	51-52	SAVING
98- 90	59-5A	VERIFYING
99-100	63-64	FOUND
106-107	6A-6B	O.K.

Met sprongadres $F12F (61743) kan deze routine ook worden gebruikt zonder dat is nagegaan of de programmamodus actueel is.

SEARCHING (FOR FILENAME) uitvoeren, sprongadres: $F5AF (62895)

Bent u niet in de programmamodus, dan wordt er onmiddellijk terug gesprongen. In het andere geval wordt SEARCHING uitgevoerd. Is de lengte van de filenaam in $B7 ongelijk 0, dan wordt de routine afgebroken. Zo niet, dan verschijnt FOR op het scherm, gevolgd door de filenaam (LSB in $BB, MSB in $BC). Voorbeeld:

POKE 183,2: POKE 187,39: POKE 188,241: SYS 62895

Wijzer 187-188 ($BB-$BC) werd op de systeemmelding O.K. ($F127) gezet: SEARCHING FOR O.K. verschijnt op het scherm. Met sprongadres $F5B3 (62899) kan de routine worden gebruikt zonder dat is nagegaan of de programmamodus actueel is.

LOADING/VERIFYING uitvoeren, sprongadres $F5D2 (62930)

LOADING: $93=0, VERYFYING: $93=1.

SAVING uitvoeren, sprongadres: $F68F (63119)

Hoeft de programmamodus niet te worden gecontroleerd, ga dan naar $F693 (63123).

TOO MANY FILES, sprongadres: $F6FB (63227)

FILE OPEN, sprongadres: $F6FE (63230)

FILE NOT FOUND, sprongadres: $F701 (63233)

DEVICE NOT PRESENT, sprongadres: $F707 (63239)

NOT INPUT FILE, sprongadres: $F70A (63242)

NOT OUTPUT FILE, sprongadres: $F70D (63245)

MISSING FILE NAME, sprongadres: $F710 (63248)

ILLEGAL DEVICE NUMBER, sprongadres: $F713 (63251)

Programmaheader van band lezen, sprongadres: $F72C (63276)

Programmaheader genereren en op band zetten, sprongadres $F76A (63338)

De volgende adressen moeten de genoemde kenmerken bevatten:

$C1-$C2	(193-194)	beginadres van programma
$AE-$AF	(174-175)	eindadres van programma
$BF	(183)	aantal tekens van filenaam
$BB-$BC	(187-188)	lo/hibytewijzer op filenaam

Stel deze waarden in alvorens de routine wordt aangesproken.

Band afzoeken naar filenaam, sprongadres: $F7EA (64466)

De filenaam moet aldus worden bepaald:

$B7 (183): lengte van filenaam

Zoekt u naar het volgende programma, zet het adres dan op 0.

$BB-$BC (187-188): adres van filenaam

De computer zoekt totdat de header is gevonden, het eind van de band is bereikt (EOT-signaal) of de routine onderbroken wordt.

Bandtoets inlezen I, sprongadres $F817 (63511)

Is een bandtoets ingedrukt, dan wordt er onmiddellijk weer terug gesprongen. Zo niet, dan verschijnt PRESS PLAY ON TAPE op het scherm. Hebt u op een toets gedrukt, dan verschijnt O.K. en er wordt terug gesprongen. De STOP-toets wordt met deze routine ingelezen.

Bandtoets inlezen II, sprongadres $F82E (63534)

Deze routine leidt niet tot uitvoer, er wordt onmiddellijk terug gesprongen. Wel wordt de Y-flag gezet:

(1) toets ingedrukt Y=1
(2) toets niet ingedrukt Y=0

Bandtoets inlezen III, sprongadres $F838 (63544)

Zie versie I. In plaats van PRESS PLAY ON TAPE verschijnt PRESS RECORD & PLAY ON TAPE.

STOP-toets, sprongadres $F8D0 (63696)

Als de STOP-toets niet is ingedrukt, wordt het programma verlaten. Is dat wel het geval, dan wordt de motor uitgeschakeld, de IRQ-routine hersteld, de carry-flag gezet en pas daarna terug gesprongen. In deze routine staan twee PLA-instructies: bij een ingedrukte STOP-toets wordt het eerste terugsprongadres gewist.

Einde recordergebruik, sprongadres $FC93 (64659)

Het beeldscherm wordt weer ingesteld, de recorder wordt uitgeschakeld en de videocontroller krijgt zijn standaardwaarden terug.

Motor van recorder uitschakelen, sprongadres $FCCA (64714)

10 KERNAL

De KERNAL is een sprongtabel, die voor alle Commodore computers hetzelfde is ($FF81-$FFF3/65409-65523). $FF81 is bijvoorbeeld altijd de oproep van de videoreset. Alleen het adres, waar naartoe wordt gesprongen, verschilt. De programma's, die via deze routines lopen, kunnen op alle Commodore computers worden gebruikt. Op die manier kunt u een programma geschikt maken voor een andere machine. Bovendien zijn er legio toepassingsmogelijkheden. In de volgende opsomming staat elke keer eerst het KERNAL-adres en vervolgens het adres, waar de computer naartoe moet springen.

KERNAL-adres: $FF81 (65409)
functie: video-reset
sprongadres: $FF5B

Het beeldscherm krijgt zijn standaardwaarden weer terug.
De subroutine:

```
FF5B    JSR $E518
FF5E    LDA $D012
FF61    BNE $FF5E
FF63    LDA $D019
FF66    AND #$01
FF6B    JMP $EDDD
```

In $FF5B (66571) wordt naar een subroutine gesprongen, die

(1) de standaardwaarden van de cursor weer instelt,
(2) omschakeling van klein naar groot weer mogelijk maakt,
(3) de lengte van de toetsenbordbuffer op 10 zet,
(4) het beeldscherm wist en
(5) de cursor aan het begin van het scherm zet.

In SFF5E-$FF61 wordt nagegaan of die regel al af is, die net op het scherm is verschenen. Zo niet, dan wacht de computer tot dat het geval is. Vervolgens wordt bit 0 ($D019/653273) in adres $02A6 (678) gezet: AND #$01 wist bit 1-7, alleen bit 0 wordt 1 of 0. Daarna springt de computer naar een programma, dat nagaat of een PAL- of NTSC-televisie is aangesloten. $02A6 voert die controle uit. Is het resultaat van

PRINT PEEK(53273) AND 1

een 1, dan heeft u een PAL-versie (16421 cycli). Een 0 is een NTSC-versie (17048 cycli). In alle Europese landen (behalve Frankrijk: SECAM) is de PAL-norm standaard. In de Verenigde Staten is de NTSC-versie (voorloper van PAL) gangbaar. In de Commodore 64 zitten routines, waarmee van de ene naar de andere norm kan worden overgeschakeld.

KERNAL-adres: $FF87 (65412)
functie: CIA's initialiseren
sprongadres: $FDA3

De CIA's (IC's voor in- en uitvoer) krijgen hun standaardwaarden weer terug. Ook deze subroutine gaat na of er een PAL- of NTSC-apparaat is aangesloten en bepaalt vervolgens de cycli.

KERNAL-adres: $FF87 (65415)
functie: RAM wissen/testen
sprongadres: $FD50

Deze routine wist de zero page behalve $00-$01, page 2 en 3. De wijzer van de cassettebuffer ($B2-$B3/178-179) wordt teruggezet en begint weer bij $033C (828). Vanaf $0400 (1024) gaat na of het einde van het RAM is bereikt. Het LSB en MSB van dit adres komt in $0283-$0284 te staan. Het begin van het RAM wordt op $0800 (2084) gezet en de start van het video-RAM op $0400 (1024). Met de routine, die hierin is verpakt kan de bovengrens van het RAM gemakkelijk worden bepaald. Breng daartoe het gewenste adres in de registers $283 (X-register: LSB) en $284 (Y-register: MSB) en spring naar de subroutine op $FE27. Ook kan de bovengrens in de registers worden gelezen met een sprong naar $FE2D. X-register: LSB. Y-register: MSB.

KERNAL-adres: $FF8A (65418)
functie: I/O-bereik initialiseren
sprongadres: $FD15

De standaardwaarden van het I/O-bereik worden ingesteld.

KERNAL-adres: $FF90 (65424)
functie: I/O-vectoren initialiseren
sprongadres: $FD1A

$0314-$0333 krijgen hun normale waarden terug.

KERNAL-adres: $FF90 (65424)
functie: status instellen
sprongadres: $FE18

```
FE18     STA $9D    ; Status voor direkte modus
                    ; ($80=Direkt, $00=Programmamodus)
FE1A     LDA $90
FE1C     ORA $90
FE1E     STA $90
FE20     RTS
```

KERNAL-adres: $FF93 (65427)
functie: secundaire adres naar LISTEN sturen
sprongadres: $EDB9

Het secundaire adres wordt op de IEC-bus (seriële uitvoer) gezet en de IRQ onderbroken. Het secundaire adres moet in de accu worden overgedragen. Met LISTEN wordt een diskdrive bedoeld, die gegevens moet ontvangen.

KERNAL-adres: $FF96 (65430)
functie: secundaire adres naar TALK sturen
sprongadres: $EDC7

Deze routine functioneert als $FF93, alleen wordt het secundaire adres naar een diskdrive gestuurd die gegevens moet zenden (TALK).

KERNAL-adres: $FF99 (65433)
functie: einde van RAM instellen/halen
sprongadres: $FE25

Het einde van het RAM wordt ingesteld (carry-flag=0) of gelezen (carry-flag=1). Hoe het X- en Y-register (LSB en MSB) worden gebruikt (lezen of schrijven) hangt van de carry-flag af.

KERNAL-adres: $FF9C (65436)

functie: begin van RAM instellen/halen
sprongadres: $FE34

Als $FF99, begin van RAM in plaats van eind.

KERNAL-adres: SFF9F (65439)
functie: uitlezen van toetsenbord
sprongadres: $EA87

De toetsen worden ingelezen, gedecodeerd en het teken wordt uitgevoerd. IRQ gebruikt deze routine niet via de KERNAL, maar direkt.

KERNAL-adres: $FFA2 (65442)
functie: time out-flag voor IEC-bus instellen
sprongadres: $FE21
De time out-flag ($0285/645) wordt ingesteld:

```
FE21      STA $0285
FE24      RTS
```

De flag moet in de accu komen te staan.

KERNAL-adres: $FFA5 (65455)
functie: invoer via IEC-bus (IECIN)
sprongadres: $EE13

Er wordt een teken van de disk gehaald. Voordien moeten de routines zijn aangesproken, die de verbinding tot stand brengen.

KERNAL-adres: $FFA8 (65448)
functie: uitvoer via IEC-bus (IECOUT)
sprongadres: $EDDD

Bij ATN wordt een teken op de IEC-bus gezet. Voordien moeten als bij IECIN andere routines zijn aangesproken.

KERNAL-adres: $FFAB (65451)
functie: UNTALK zenden
sprongadres: $EDEF

Na UNTALK wordt de communicatie onderbroken met een actief in- en uitvoerapparaat, dat gegevens naar de computer stuurt.

KERNAL-adres: $FFAE (65454)
functie: UNLISTEN zenden
sprongadres: $EDFE

Als UNTALK, alleen wordt een apparaat aangesproken, dat gegevens ontvangt.

KERNAL-adres: $FFB1 (65457)

functie: LISTEN zenden
sprongadres: $ED0C

De tegenhanger van UNLISTEN ($FFAE): hier begint de communicatie met een apparaat. Het nummer van de diskdrive moet voordien in de accu worden geladen en opgeslagen in $BA (186). Wilt u gegevens naar een diskdrive sturen, dan moeten de eerste drie regels er zo uitzien:

```
LDA  #$08       ; 8=Device-nummer van diskdrive
STA  $BA        ; Opslaan
JSR  $FFB1      ; LISTEN-routine
```

KERNAL-adres: $FFB4 (65460)
functie: TALK zenden
sprongadres: $ED09

Als LISTEN, alleen wordt hier aan de drive kenbaar gemaakt, dat hij gegevens moet zenden. TALK en LISTEN gebruiken dezelfde routine:

```
ED09 ORA #$40
ED0B .BYTE $2C
ED0C ORA #$20
ED0E JSR $F0A4
```

In $ED09 staat de sprong voor TALK, in $ED0C die voor LISTEN. Bij TALK worden het apparaatnummer en 64 door de logische operator OR met elkaar vergeleken: behalve het devicenummer, dat bit 0-3 in beslag kan nemen, wordt bit 6 ingesteld. Vervolgens wordt $ED0C overgeslagen, omdat anders ook bit 5 nog zou worden gezet. Bij LISTEN worden het devicenummer en 32 door de logische operator OR vergeleken: bit 5 wordt ingesteld. Vanaf $ED0E doen beide routines hetzelfde. Bij een ingestelde bit herkennen ze dat de drive:

(1) via bit 5 van het devicenummer gegevens ontvangt (LISTEN)
(2) via bit 6 van het devicenummer gegevens zendt (TALK)

KERNAL-adrse: $FFB7 (65463)
functie: status halen
sprongadres: $FE07

De status in de accu wordt gehaald en op 0 gezet. Ook voor het RS-232 interface kan de routine worden gebruikt, als het aparaatnummer ($BA/186) =2.

KERNAL-adres: $FFBA (65466)

functie: fileparameters instellen
sprongadres: $FE00

Devicenummer (primair adres), secundair adres en het logische filenummer moeten aan de subroutine worden doorgegeven.

PARAMETERS	REGISTER	WORDT OPGESLAGEN IN
Logisch filenummer	accu	$B8 (184)
Apparaatnummer	X-register	$BA (186)
Secundair adres	Y-register	$B9 (185)

KERNAL-adres: $FFBD (65469)
functie: parameters voor filenaam instellen
sprongadres: $FDF9

PARAMETERS	REGISTER	WORDT OPGESLAGEN IN
Lengte van naam	accu	$B7 (183)
Adres low	X-register	$BB (187)
Adres high	Y-register	$BC (188)

Net als bij KERNAL-routine $FFBA moeten de waarden in de accu, het X-register en het Y-register worden overgedragen.

KERNAL-adres: $FFC0 (65472)
functie: OPEN
sprongadres: volgt uit $031A-$031B; gewoonlijk $F34A

De listing van deze belangrijke instructie:

```
F34A  LDX $B8      ; laadt logisch filenummer
F34C  BNE $F351    ; ongelijk nul, dan verder
F34E  JMP $F70A    ; voert "not output file" uit
F351  JSR $F30F    ; logisch filenummer al aanwezig?
```

```
F354  BNE $F359     ; nee
F356  JMP $F6FE     ; voert "file open" uit
F359  LDX $98       ; aantal open files
F35B  CPX #$0A      ; met 10 vergelijken
F35D  BCC $F362     ; kleiner dan 10, verder
F35F  JMP $F6FB     ; te groot, dus "too many files"
F362  INC $98       ; aantal met 1 verhogen
F364  LDA $B8       ; logisch filenummer
F366  STA $0259,X   ; slaat deze op in tabel
F369  LDA $B9       ; secundair adres
F36B  ORA #$60      ; bit 5 en 6 voor diskdrive instellen
F36D  STA $B9       ; weer opslaan
F36F  STA $026D,X   ; secundair adres in tabel opslaan
F372  LDA $BA       ; devicenummer
F374  STA $0263,X   ; in tabel opslaan
F377  BEQ $F3D3     ; toetsenbord, dus return
F379  CMP #$03      ; beeldscherm?
F37B  BEQ $F3D3     ; ja, dus return
F37D  BCC $F384     ; geen file op iec-bus
F37F  JSR $F3D5     ; file op iec-bus openen
F382  BCC $F3D3     ; klaar
F384  CMP #$02      ; rs-232?
F386  BNE $F38B     ; nee, dus band
F388  JMP $F409     ; rs-232 open
F38B  JSR $F7D0     ; cassettebuffer-begin halen
F38E  BCS $F393     ; band-open verder
F390  JMP $F713     ; voert "illegal device number" uit
F393  LDA $B9       ; secundair adres
F395  AND #$0F      ; bits 4-7 wissen
F397  BNE $F3B8     ; secundair adres; 0, schrijven
F399  JSR $F817     ; play-toets inlezen
F39C  BCS $F3D4     ; stop-toets ingedrukt, stoppen
F39E  JSR $F5AF     ; "searching for name" uitvoeren
F3A1  LDA $B7       ; lengte van filenaam
F3A3  BEQ $F3AF     ; geen filenaam, verder
F3A5  JSR $F7EA     ; zoekt gewenste tapeheader
F3A8  BCC $F3C2     ; gevonden
F3AA  BEQ $F3D4     ; stop
F3AC  JMP $F704     ; eot, dus "file not found"
F3AF  JSR $F72C     ; verder zoeken
F3B2  BEQ $F3D4     ; eot, dus einde
F3B4  BCC $F3C2     ; gevonden
F3B6  BCS $F3AC     ; verder zoeken
F3B8  JSR $F838     ; wacht op record&play-toets
F3BB  BCS $F3D4     ; stop-toets ingedrukt, dan stoppen
F3BD  LDA #$04      ; controlebyte voor tapeheader
F3BF  JSR $F76A     ; header op band schrijven
F3C2  LDA #$BF      ; wijzer op einde van cassettebuffer
```

```
F3C4  LDY $B9        ; secundair adres
F3C6  CPY #$60       ; met 96 (bit 5 & 6) vergelijken
F3C8  BEQ $F3D1      ; secundair adres is nul, dus verder
F3CA  LDY #$00
F3CC  LDA #$02       ; controlebyte voor gegevensblok
F3CE  STA ($B2),Y    ; in cassettebuffer schrijven
F3D0  TYA            ; accu gelijk
F3D1  STA $A6        ; wijzer op cassettebuffer
F3D3  CLC
F3D4  RTS
F3D5  file op iec-bus openen
```

Het instellen van de volgende parameters dient aan de OPEN-routine vooraf te gaan:

VOOR FILENAAM:

(1) lengte ($B7/183)
(2) adres low ($BB/187)

(3) adres high ($BC/188)

VOOR FILE

(1) logisch filenummer ($B8/184)
(2) secundaire adres ($B9/185)
(3) devicenummer ($BA/186)

Deze parameters worden ingesteld door twee routines in de KERNAL:

(1) $FFBA fileparameters instellen
(2) $FFBD parameters voor filenaam instellen

Voordat de OPEN-routine wordt opgeroepen, moet er naar deze programma's worden gesprongen. Als voorbeeld de opening van een directory (naam: $) op de disk:

```
LDA #$01  ; LOGISCH FILENUMMER
LDX #$D0  ; ADRES LOW
LDY $FF   ; ADRES HIGH
JSR $FFBD ; FILENAAMPARAMETER BEPALEN
-------------------------------------
LDA #$01  ; LOGISCH FILENUMMER
LDY #$00  ; SECUNDAIR ADRES
LDX #$08  ; DEVICE-NUMMER
JSR $FFBA ; FILEPARAMETER BEPALEN
-------------------------------------
JSR $FFC0 ; OPEN
```

$FFDO is het adres van de filenaam. Op deze geheugenplaats staat 36, de ASCII-waarde van $.

KERNAL-adres: $FFC3 (65475)
functie: CLOSE
sprongadres: bepaald door vector $031C-$031D (796-797),

gewoonlijk $F291

Alleen het logische filenummer moet worden aangegeven. In BASIC staat ook alleen maar "CLOSE 1", als de file met het logische filenummer 1 moet worden afgesloten. Voordat de OPEN-routine wordt aangesproken, moet het logische filenummer van de file die u wilt afsluiten, in de accu zijn opgeslagen. Bij een file met nunmmer 10 gaat dat in machinetaal als volgt:

```
LDA  #$0A
JSR  $FFC3
RTS
```

Hierover valt niets te zeggen. Voor de gebruiker is alleen belangrijk dat het programmaatje functioneert. De computer houdt het aantal open files ($98/158) bij. De andere waarden staan in tabellen:

```
601-610  tabel voor logische files
611-620  tabel voor devicenummers
621-630  tabel voor secundaire adressen
```

In het begin zijn alle waarden in de tabellen 0. Wordt een file met nummer 1, secundair adres 0 en apparaatnummer 8 geopend, dan wordt automatisch het aantal logische files met 1 verhoogd, zodat in adres $98 (158) een 1 staat. De drie tabellen zien er daarna zo uit:

	1	2	3	4	5	6	7	8	9	10	
601	1	0	0	0	0	0	0	0	0	0	filenummer
611	0	0	0	0	0	0	0	0	0	0	secundair adres
621	8	0	0	0	0	0	0	0	0	0	devicenummer

Er wordt nog een file geopend; logisch nummer 3, secundair adres 1, apparaatnummer 1:

	1	2	3	4	5	6	7	8	9	10	
601	1	3	0	0	0	0	0	0	0	0	filenummer
611	0	1	0	0	0	0	0	0	0	0	secundair adres
621	8	1	0	0	0	0	0	0	0	0	apparaatnummer

Voor de andere waarden van file 3 raadpleegt de computer eerst tabel 601-610. Per plaats wordt een teller 1 hoger. Heeft de computer het logische filenummer gevonden, dan onthoudt hij de waarde van de wijzer. Deze waarde geldt ook voor andere tabellen: de file staat op dezelfde plaats.

KERNAL-adres: $FFC6 (65478)
functie: invoerapparaat instellen
sprongadres: bepaald door vector $031E-$031F (798-799)
gewoonlijk $F20E

Het nummer van de file die het invoerapparaat bepaalt, moet in het X-register worden gezet. Als de file niet is geopend, verschijnt FILE NOT OPEN op het scherm. Het apparaatnummer wordt in $99 (153) geschreven. Het aanspreken van de diskdrive, dus de IEC-bus, wordt gevolgd door een TALK-signaal: de status wordt getest. Antwoordt het apparaat niet, dan wordt DEVICE NOT PRESENT uitgevoerd. Is de datasette het invoerapparaat (nummer 1) en is het secundaire adres iets anders dan 0 (lezen), dan verschijnt NOT INPUT FILE op het scherm, omdat bijvoorbeeld een output file (secundaire adres 1=schrijven) geen invoerapparaat kan bepalen. Wilt u de routine gebruiken zonder dat er een file is geopend, zorg er dan voor dat apparaatnummer en secundair adres al in $BA (186) en $B9 (187) staan. Het sprongadres is in dat geval $F219 (61977). Het nummer van het invoerapparaat komt in $99 (153).

KERNAL-adres: $FFC9 (65481)
functie: uitvoeraparaat instellen (CKOUT)
sprongadres: bepaald door vector $0320-$0321 (800-801)
gewoonlijk $F250

Het logische filenummer moet in het X-register komen te staan. Als de file niet open is, komt dat op het scherm: FILE NOT OPEN. Bepaalt een file, die door de datasette wordt aangesproken, het uitvoerapparaat, maar is het secundaire adres 0 (lezen), dan verschijnt NOT OUTPUT FILE. Ook met deze routine kan een uitvoerapparaat worden ingesteld zonder dat een file geopend is. Zet devicenummer en secundaire adres in $BA (186) en $B9 (185) en gebruik $F25B (62043) als sprongadres. Bij het aanspreken van de diskdrive wordt LISTEN uitgestuurd, de status uitgelezen en

eventueel DEVICE NOT PRESENT uitgevoerd. Het nummer van het uitvoerapparaat komt in $9A (154) te staan.

```
KERNAL-adres: $FFCC (65484)
     functie: I/O-reset (CLRCH)
 sprongadres: bepaald door vector $0322-$0323 (802-803)
              gewoonlijk $F333
```

Een actief I/O-kanaal wordt verbonden met de IEC-bus:

```
F333  LDX #$03      ; 3=beeldscherm
F335  CPX $9A       ; nummer van uitvoerapparaat
F337  BCS $F33C     ; kleiner dan 3
F339  JSR $EDFE     ; unlisten naar diskdrive sturen
F33C  CPX $99       ; nummer van invoerapparaat
F33E  BCS $F343     ; kleiner dan 3
F340  JSR $EDEF     ; untalk naar diskdrive sturen
F343  STX $9A       ; uitvoer weer op scherm
F345  LDA #$00      ; 0=toetsenbord
F347  STA $99       ; invoer weer van toetsenbord
F349  RTS
```

Als de uitvoer op de IEC-bus is gezet, wordt er naar een routine gesprongen die de gegevensstroom opschort (UNLISTEN). Is de diskdrive het uitvoerapparaat dan geeft het UNTALK-signaal aan, dat de drive geen gegevens meer naar de computer moet zenden. Bij beide mogelijkheden (uitvoer naar disk; invoer vanaf disk) verschijnt de uitvoer weer op het scherm en loopt de invoer via het toetsenbord (standaardmodus).

```
KERNAL-adres: $FFCF (65487)
     functie: invoer van teken (BASIN)
 sprongadres: bepaald door vector $0324-$0325 (804-805)
```

Na de sprong terug in het hoofdprogramma staat het teken in de accu. Een string die met het toetsenbord is ingelezen, moet worden opgeslagen in het bereik vanaf $033C. Het programma slaan we op vanaf $6000 (24576).

```
  LDX #$00       ; TELLER OP NUL
* JSR $FFCF      ; BASIN(TEKEN HALEN)
  STA $033C,X    ; TEKEN OPSLAAN
  INX            ; TELLER VERHOGEN
  CMP #$0D       ; RETURN?
  BNE *          ; NEE
  RTS            ; JA
```

In BASIC:

```
10 FOR Y=0 TO 13: READ A:POKE24576+Y,A:NEXT Y: SYS24576
20 DATA 162,0,32,207,255,157,60,3,232,201,13,208,245,96
```

Na RUN verschijnt een cursor op het scherm: u kunt uw tekens invoeren. Met RETURN wordt het programma afgebroken. Controleer of de string goed is weggeschreven. L, de lengte van de string, mag niet groter zijn dan 192, anders is de cassettebuffer vol.

```
FOR Y=0 TO L:PRINTCHR$(PEEK(828+Y));:NEXT Y
```

De tekens (ook cursorfuncties) verschijnen op het scherm. Maar de routine is niet beperkt tot het toetsenbord. Ook gegevens van band, disk of RS-232 interface kunnen worden overgenomen, als het invoerapparaat is veranderd.

OVERNAME VAN GEGEVENS VAN DE DISK

```
(1) parameters instellen  LDA #$01 ; logisch filenummer
                          LDX #$08 ; devicenummer
                          LDY #$00 ; secundair adres
                          JSR $FFBA; parameters instellen

(2) parameters voor       LDA #$xx ; lengte van naam
    filenaam instellen    LDX #$y1 ; adres low
                          LDY #$y2 ; adres high
                          JSR $FFBD; parameters instellen

(3) OPEN                  JSR $FFC0; OPEN-routine

(4) invoerapparaat        LDX #$01 ; logisch filenummer
    instellen             JSR $FFC6; CHKIN

(5) gegevens halen uit               BASIN
    geopende file

(6) zijn niet alle gegevens overgenomen, dan (5)

(7) terug naar            JSR $FFCC; CLRCH
    standaardwaarden

(8) file sluiten          LDA #$01 ; logisch filenummer
                          JSR $FFC3; CLOSE
```

Dit systeem kan voor alle soorten files worden gebruikt. Een voorbeeld vindt u in paragraaf 2.21.

```
KERNAL-adres: $FFD2 (65490)
     functie: uitvoer van teken (BSOUT)
 sprongadres: bepaald door vector $0326-$0327 (806-807)
              gewoonlijk $F1CA
```

Een voorbeeld:

```
     LDX #$00        ; Teller op nul
   * LDA $F0BE,X     ; Teken lezen
     JSR $FFD2       ; en uitvoeren(BASOUT)
     INX             ; Teller verhogen
     CPX #$09        ; al 9?
     BNE *           ; nee
     RTS             ; ja
```

De routine zet de adressen $F0BE-$F0C6 (61630-61638 (I/O ERROR)) een voor een in de accu. Door BSOUT verschijnen ze op het scherm. In BASIC:

```
10 FOR X=0 TO 13: READ A: POKE24576+X,A: NEXT X: SYS24576
20 DATA 162,0,189,190,240,32,210,255,232,224,9,208,245,96
```

Op dezelfde manier als bij de BASIN-routine kan BSOUT voor andere apparaten worden gebruikt. Zorg er wel voor, dat

(1) het secundaire adres 1 is
(2) CKOUT wordt aangesproken: het betreft een uitvoerapparaat
(3) BSOUT wordt aangesproken

```
KERNAL-adres: $FFD5 (65493)
     functie: LOAD
 sprongadres: $F49E
```

Voordat de LOAD-routine wordt aangesproken, moeten de volgende parameters zijn ingesteld:

$BA (186)	devicenummer
$B7 (183)	lengte van filenaam; noodzakelijk bij diskdrive, mag anders 0 zijn
$B9 (185)	secundaire adres
$BB-$BC	LSB/MSB op filenaam; niet nodig als $B7=0
X-register	LSB van beginadres (komt in $C3 (195))
Y-register	MSB van beginadres (komt in $C4)
accu	LOAD/VERIFY-flag (0=LOAD, 1=VERIFY) ($93 (147))

In $F4A2 van de LOAD-routine staat een indirekte sprong:

```
JMP ($0330)
```

Meestal levert dat $F4A5 op. Deze vector kan ook op andere LOAD-routines worden gezet (zie hoofdstuk 1).

KERNAL-adres: $FFD8 (65496)
functie: SAVE
sprongadres: $F5DD

De waarden voor de SAVE-routine:

$AE-$AF	eindadres
$C1-$C2	beginadres
$BA	devicenummer
$B9	secundair adres (wordt bij IEC-SAVE vanzelf 1)
$B7	lengte van filenaam (bij IEC-SAVE 0 niet toegestaan)
$BB-$BC	adres van filenaam (n.v.t. als $B7=0)

In een lus worden alle bytes naar het apparaat gebracht.

KERNAL-adres: $FFDB (65499)
functie: tijd instellen
sprongadres: $F6E4

De accu, het X- en het Y-register stellen de tijd in. De tijd staat in $A0-$A2 (160-162):

```
TI=PEEK(160)*65525+PEEK(161)*256+PEEK(162)
```

De accu komt in $A2 (LSB van klok)
Het X-register komt in $A1 (mid-byte van klok)
Het Y-register komt in $A0 (MSB van klok)

KERNAL-adres: $FFDE (65502)
functie: tijd halen
sprongadres: $F6DD

Na de sprong terug in het hoofdprogramma hebben de registers de volgende waarden:

REGISTER	ADRES	FUNCTIE
accu	$A2 (162)	lo-byte van klok
X-reg.	$A1 (161)	mid-byte van klok
Y-reg.	$A0 (160)	hi-byte van klok

```
KERNAL-adres: $FFE1 (65505)
     functie: STOP-toets inlezen
 sprongadres: bepaald door vector $0328-$0329 (808-809)
              gewoonlijk $F6ED
```

Deze routine functioneert alleen met behulp van IRQ. In IRQ wordt naar een subroutine gesprongen, die een flag zet ($91 (145)) als STOP is ingedrukt. Is de inhoud van de flag die de subroutine in de KERNAL inleest, $7F (127), dan wordt de KERNAL-routine CLRCH ($FFCC) uitgevoerd. De toetsenbordbuffer wordt leeg gemaakt, doordat een 0 in $C6 (198) komt te staan. Als POKE 788,52 de STOP-toets heeft uitgeschakeld, werkt de routine niet meer tenzij u vanuit IRQ eerst naar de subroutine springt en dan pas naar het programma in de KERNAL.

```
KERNAL-adres: $FFE4 (65508)
     functie: GETIN
 sprongadres: bepaald door vector $032A-$032B (810-811)
              gewoonlijk $F13E
```

Dit programma haalt een teken. Via $99 (153: actief invoerapparaat) springt de computer naar verschillende routines:

$99 (153)	INVOERAPPARAAT	SPRONGADRES
0	toetsenbord	$E5B4
2	RS-232 interface	$F086
anders	disk, band etc.	$F166

Als een toets wordt ingelezen, ziet de routine er als volgt uit:

```
E5B4  LDY $0277    ; eerste teken in y-register
E5B7  LDX #$00     ; teller
E5B9  LDA $0278,X  ; teken halen
E5BC  STA $0277,X  ; en opslaan op voorgaand adres
E5BF  INX          ; teller verhogen
E5C0  CPX $C6      ; aantal tekens
E5C2  BNE $E5B9    ; nog niet allemaal, dus verder
E5C4  DEC $C6      ; aantal tekens met 1 verlagen
E5C6  TYA          ; eerste teken in accu halen
E5C7  CLI          ; interrupt weer normaal
E5C8  CLC
E5C9  RTS
```

Dit programma haalt het eerste teken uit de toetsenbordbuffer (631-640), schuift de andere tekens 1 op en vermindert het getal met een teken. De interrupt werd verhinderd om uit te sluiten dat de toetsenbordbuffer voor korte tijd zou worden gewijzigd. De IRQ-routine leest het toetsenbord uit (zie hoofdstuk 7).

KERNAL-adres: $FFE7
functie: alle open files sluiten (CLALL)
sprongadres: bepaald door vector $032C-$032D (812-813) gewoonlijk $F32F

Dit programma gebruikt de KERNAL-routine CLRCH ($FFCC). Met

```
$F32F  LDA #$00
$F331  STA $98
```

worden alle open files gesloten ($98 (158) geeft aantal open files). Vanaf $F333 volgt de CLRCH-routine.

KERNAL-adres: $FFEA (65514)
functie: tijd verhogen/STOP-toets inlezen
sprongadres: $F69B

Deze routine is ook van IRQ afhankelijk, maar $F69B wordt direkt aangesproken. Is de waarde voor 24 uur bereikt, dan komt de klok weer op 0 te staan. Als de STOP-toets is ingedrukt, wordt flag $91 (145) gezet. In IRQ wordt alleen de STOP-toets ingelezen, er wordt niet naar een routine gesprongen. Dat is een voordeel. Wordt STOP ingedrukt in de laadmodus, dan wordt een andere routine aangesproken dan in de programmamodus.

KERNAL-adres: $FFED (65517)
functie: aantal regels/kolommen halen
sprongadres: $E505

Het aantal kolommen (40) gaat naar het X-, de regels (25) naar het Y-register:

```
$E505  LDX #$28  ; aantal kolommen
$E507  LDY #$19  ; aantal regels
$E509  RTS
```

KERNAL-adres: $FFF0 (65520)
functie: cursor instellen/halen
sprongadres: $E50A

Is de carry flag gezet, dan wordt de cursorpositie gehaald, anders ingesteld.

```
E50A BCS $E513      ; instellen
E50C STX $D6        ; Regel
E50E STY $D3        ; Kolom
E510 JSR $E56C      ; Cursor instellen
E513 LDX $D6
E515 LDY $D3
E517 RTS
```

Wordt de cursorpositie gehaald, dan staat de regel in het X- en de kolom in het Y-register. Wordt die ingesteld, dan moeten regel en kolom in die registers worden overgedragen.

KERNAL-adres: $FFF3
functie: beginadres van I/O-bouwsteen halen
sprongadres: $E500

Is de routine aangesproken, dan staat de LSB van het beginadres van de I/O-bouwsteen in het X-register en de MSB ($DC) in Y.

```
$E500  LDX #$00; LSB
$E502  LDY #$DC; MSB
$E504  RTS
```

Deze routine en de routine die het aantal regels en kolommen haalt, zijn nodig als u programma's op andere computers wilt gebruiken. Met deze routines kunnen programma's zo worden geschreven, dat ze vanzelf het beeldschermformaat van de computer opvragen.

11 HET GEHEUGEN

11.1 Het opslaan van BASIC-regels

Omdat het BASIC-geheugen bij 2049 begint, moet in 2048 een nul staan. Poken we in dit adres een 1, dan geeft de computer bij NEW of RUN de foutmelding SYNTAX ERROR (pas op, NEW wordt wel uitgevoerd). Het programma in deze paragraaf laat zien hoe de computer BASIC-regels opslaat. De bovenste regel van het scherm geeft het aantal bytes van het BASIC-geheugen aan, dat de laatste regel heeft gebruikt. Na RUN verschijnt

    00 BYTES GEBRUIKT

De laatste invoer (na de laatste RETURN) heeft geen geheugenruimte in beslag genomen. Alleen programmaregels gebruiken geheugenruimte. Na

    100 PRINT

staat er

    06 BYTES GEBRUIKT

De eerste 5 bytes zijn voor het beheer, de laatste byte is voor de instructiecode (token) voor PRINT:

byte 1-2	LSB en MSB van beginadres van volgende BASIC-regel. De laatste regel wijst op 0.
byte 3-4	LSB en MSB van regelnummer. 1000 levert 232-233 op.
byte 5	Eind van regel. De waarde moet 0 zijn.

    20 PRINT"CASPER"

gebruikt 14 bytes: 5 voor het beheer, 1 voor het token (PRINT) en de overige 8 voor de naam en de aanhalingstekens.

    30 PRINTCHR$(48)

gebruikt 11 bytes: 5 voor het beheer, 1 voor het BASIC-token 'PRINT', 1 voor het BASIC-token 'CHR$' en 4 voor '(48)'. De functies TAB en SPC bevatten al het teken '('.

```
40 POKE 198,0
```

bestaat uit 5 bytes voor het beheer, 1 voor 'POKE' en 5 voor de tekst '198,0'. Het programma in BASIC:

```
10 FOR I=0 TO 87
20 READ A
30 POKE 40704+I,A
40 S=S+A
50 NEXT I
60 IF S<>7544 THEN PRINT "FOUT IN DATA":END
70 PRINT"DATA O.K."
80 SYS 40704
100 DATA 120,169,15,141,20,3,169,159,141,21,3,133,56,88,96,169,48
110 DATA 141,0,4,141,1,4,165,11,201,76,240,18,56,201,10,144,8,238
120 DATA0,4,233,10,76,30,159,105,48,141,1,4,162,2,189,69,159,157,0,4
130 DATA232,224,18,208,245,169,1,202,157,0,216,208,250,76,49,234
140 DATA32,2,25,20,5,19,32,7,5,2
150 DATA18,21,9,11,20,32,32
```

In machinetaal:

```
9F00   SEI
9F01   LDA #0F
9F03   STA $0314
9F06   LDA #9F
9F08   STA $0315
9F0B   STA $38
9F0D   CLI
9F0E   RTS
9F0F   LDA #$30
9F11   STA $0400
9F14   STA $0401
9F17   LDA $0B
9F19   CMP #$4C
9F1B   BEQ $9F2F
9F1D   SEC
9F1E   CMP #$0A
9F20   BCC $9F2A
9F22   INC $0400
```

```
9F25   SBC #$10
9F27   JMP $9F1E
9F2A   ADC #$30
9F2C   STA $0401
9F2F   LDX #$02
9F31   LDA $9F45,X
9F34   STA $0400,X
9F37   INX
9F38   CPX #$12
9F3A   BNE $9F31
9F3C   LDA #$00
9F3E   DEX
9F3F   STA $D800,X
9F42   BNE $9F3E.
9F44   JMP $EA31
```

De rest van de gegevens zijn de beeldschermcodes voor de string BYTES GEBRUIKT.

11.2 De BASIC-monitor

Met het programma in deze paragraaf kan in het geheugen worden gekeken. U kunt BASIC-programma's tevoorschijn halen of delen van het ROM onderzoeken. Dit programma doet hetzelfde als de instructie M (memory display) voor de machinetaalmonitor. De computer vraagt eerst naar begin- en eindadres. Op het scherm verschijnen het beginadres en de PEEKs van dit adres en de vijf volgende. Aan het eind van de regel staan de beeldschermcodes van deze geheugeninhouden.

```
2048   16  15  48  61  127  102        100=

       |""""""""""""""""""""""""""""""""2048""|||||
            |"""""""""""""""""""""""""" 2049"""||||
                 |""""""""""""""""""""""2050""""|||
                      |"""""""""""""""""2051"""""||
                           |""""""""""""2052""""""|
                                |""2053"""""""""""|
```

Op het scherm staat het patroon van de regel die begint met 2048. Nu is te zien welke tekens met welke getallen zijn verbonden. Er worden decimale getallen uitgevoerd, omdat de monitor BASIC-programma's moet onderzoeken. Is de BASIC-monitor gestart, dan meldt de computer zich met

BASIC-MONITOR

BEGINADRES?

Lees het adres in (decimaal getal).

EINDADRES?

Lees het gewenste adres in. De waarden verschijnen op scherm volgens het bovenstaande patroon. De computer voert in de laatste regel 6 waarden uit, ook als het eindadres er minder aangeeft. Met F1 kan de routine worden onderbroken. Twee belangrijke beginadressen zijn: 2048 (begin van BASIC-geheugen; meestal de plaats van de monitor) en 40964 (BASIC-instructies). De monitor:

```
10 print"██"chr$(14):poke 53281,6:poke 5
3280,6
20 print" █            Basic-Monitor aan
     █":print:print
30 input" Beginadres  ";b
40 input" Eindadres   ";e
50 for i=a to e step 6
60 print i;
70 for o=0 to 5
80 xp=peek(i+o):gosub 200: rem print usi
```

```
n9
90 u=peek(214)*40+1024+33+o
100 pokeu,peek(i+o):poke54272+u,1
110 next o
120 print
130 nexti
140 input" ▀       nog een keer (j/n)
   "
150 geta$:ifa$="" then 150
160 if a$="j" then run
170 end
190 :
200 rem print formateren
210 xp$=str$(xp)
220 if len(xp$)<4 then xp$=" "+xp$:goto
220
240 print xp$;:return
```

De monitor is in BASIC geschreven, maar als u een paar waarden verandert, kunnen er ook BASIC-programma's mee worden onderzocht. Zet de computer in de inschakelmodus en geef

POKE 44,150

In plaats van 150 kunt u een ander getal nemen tussen 8 en 159. De BASIC-start wordt verhoogd, zodat monitor en het programma dat moet worden onderzocht, elkaar niet overlappen.

POKE 256*PEEK(44),0

zet een noodzakelijke 0 aan het begin van BASIC. Laadt de BASIC-monitor.

POKE 44,8

geeft de BASIC-start de normale warden terug. Laadt vervolgens de routine, die moet worden onderzocht. Met

POKE 44(uw waarde tussen 8 en 159)

komt u weer bij de monitor. Nadat de monitor is gestart, kan het programma worden onderzocht (vanaf 2048). Door de monitor met andere instructies uit te breiden, kunnen BASIC-programma's worden gewijzigd.

11.3 De zero page

Met behulp van de volgende listing kan aan onbekende adressen in vreemde listings een betekenis worden toegekend.

HEX.	DEC.		FUNCTIE
$0000	0		6510 datarichtingsregister processorpoort
$0001	1	055	6510 processorpoort
			bit 0: RAM of ROM ($A000-$BFFF)
			(BASIC-interpreter)*POKE 1,54
			bit 1: RAM of ROM ($E000-$FFFF)
			(bedrijfssysteem, KERNAL)*POKE 1,53
			bit 2: I/O of ROM ($D000-$DFFF)
			(kar.gen. aanspreken)*POKE 1,51
$0002	2		buiten gebruik
$0003	3		vector low (reëel wordt integer)
$0004	4		vector high
$0005	5		vector low (integer wordt reëel)
$0006	6		vector high
$0007	7		zoekteken
$0008	8		flag (stringmodus)
$0009	9		geheugen voor kolom bij TAB-instructie
$000A	10		flag (0=LOAD, 1=VERIFY)
$000B	11		wijzer (invoerbuffer); aantal elementen
$000C	12		flag (standaard-DIM)
$000D	13		type-flag ($FF(255)=string, 00=numeriek)
$000E	14		type-flag ($80(128)=integer, 00=reëel)
$000F	15		flag (DATA/LIST); stringflag bij LIST
$0010	16		flag (FNx, element)
$0011	17		invoerflag (00=INPUT, 64=GET, 136=READ)
$0012	18		voorteken bij ATN (flag voor gelijkheid
			bij vergelijking)
$0013	19		actief I/O-apparaat
			POKE19,1:INPUT"ZONDER VRAAGTEKEN";A$:POKE19,0
			(vraagteken uitschakelen bij INPUT)
$0014	20		integer-adres low
$0015	21		integer-adres high
$0016	22		wijzer op tijdelijke stringstack
			POKE 22,35: LIST
			(Listen zonder regelnummers)
$0017	23		vector low (laatste tijdelijke string)
$0018	24		vector high
$0019	25		stack voor tijdelijke strings
$001A	26		idem
$001B	27		idem
$001C	28		idem
$001D	29		idem
$001E	30		idem

$001F	31		idem
$0020	32		idem
$0021	33		idem
$0022	34		bereik voor hulpwijzers (voor mach.prog.)
$0023	35		idem
$0024	36		idem
$0025	37		idem
$0026	38		bereik voor produkt bij vermenigvuldiging
$0027	39		idem
$0028	40		idem
$0029	41		idem
$002A	42		idem
$002B	43	01	vector low (BASIC-start) op nieuwe BASIC-start POKE(newadd)-1,0:NEW!!
$002C	44	08	vector high
$002D	45		vector low (begin van variabelen)
$002E	46		vector high
$002F	47		vector low (begin van arrays (velden))
$0030	48		vector high
$0031	49		vector low (einde van array)
$0032	50		vector high
$0033	51		vector low (begin van strings)
$0034	52		vector high
$0035	53		vector low (hulpwijzers voor strings)
$0036	54		vector high
$0037	55		vector low (einde van BASIC-geheugen) BASIC-geheugen verlagen, bijvoorbeeld om te voorkomen dat over daar opgeslagen assemblerprogramma's heen wordt geschreven.
$0038	56		vector high
$0039	57		LSB (actuele regelnummer) PRINT PEEK(57)+256*PEEK(58) geeft actuele nummer van BASIC-regel (alleen in programmamodus)
$003A	58		MSB
$003B	59		LSB (vorige regelnummer) PRINT PEEK(59)+256*PEEK(60) geeft ook in de direkte modus het nummer van de vorige BASIC-regel
$003C	60		MSB
$003D	61		LSB (volgende instructie voor CONT). wijzer op sprongadres bij CONT
$003E	62		MSB
$003F	63		LSB (nummer van actuele DATA-regel) In verband met READ (kan worden gebruikt voor de uitvoer van regelnummers bij eigen foutmeldingen)
$0040	64		vector high

$0041	65		vector low (adres van actuele DATA-element) Wijst op beginadres van volgend DATA-element in BASIC-geheugen
$0042	66		vector high
$0043	67		vector low (wijzer op herkomst van invoer)
$0044	68		vector high
$0045	69		REG 1 variabelenaam (zie paragraaf 11.5)
$0046	70		REG 2 idem
$0047	71		vector low (adres van variabele)
$0048	72		vector high
$0049	73		LSB (waarde van variabele)
$004A	74		MSB
$004B	75		LSB (tussengeheugen voor programmawijzer)
$004C	76		MSB
$004D	77		masker voor vergelijkingsoperaties
$004E	78		vector low (wijzer voor FN)
$004F	79		vector high
$0050	80		stringdescriptor
$0051	81		idem
$0052	82		idem
$0053	83		idem
$0054	84	76	constante $4C (JMP voor functies)
$0055	85		sprongvector voor functies (low)
$0056	86		sprongvector voor functies (high)
$0057	87		rekenregister, accu 3
$0058	88		idem
$0059	89		idem
$005A	90		idem
$005B	91		idem
$005C	92		rekenregister, accu 2
$005D	93		idem
$005E	94		idem
$005F	95		idem
$0060	96		idem
$0061	97		accu voor reële getallen 1, FAC
$0062	98		idem
$0063	99		idem
$0064	100		idem
$0065	101		idem
$0066	102		voorteken van FAC
$0067	103		wijzer voor berekening van veelterm (bijvoorbeeld vanaf $E059)
$0068	104		afrondingsbyte voor FAC
$0069	105		accu voor reële getallen 2, ARG
$006A	106		idem
$006B	107		idem
$006C	108		idem
$006D	109		idem
$006E	110		voorteken van ARG

$006F	111	voortekens van FAC en ARG vergelijken
$0070	112	afrondingsbyte voor FAC

De routine die formules interpreteert (BASIC-interpreter vanaf $AD9E), werkt met reële getallen. Integere variabelen (hele getallen) worden eerst getransformeerd. Het programma kan ook stringparameters verwerken. Een type-flag (adres 13) onderscheidt stringparameters van andere numerieke variabelen. Numerieke waarden worden tussentijds in FAC 1 (vanaf $61) opgeslagen. Omdat deze accu niet voldoende is voor rekenkundige operaties, is er nog een accu voor reële getallen, ARG (vanaf $69). Argument en resultaat staan in de FAC. Bij de interpretatie van strings vormen $64-$65 een wijzer op de stringdescriptor. Daar staan de lengte en het adres van de bewerkte strings. De volgende interpreterroutines zijn belangrijk voor de interpretatie van termen bij programmering in machinetaal.

```
$B86A (47210)  FAC=ARG+FAC
$B853 (47187)  FAC=ARG-FAC
$BA28 (47656)  FAC=ARG*FAC
$BB12 (47890)  FAC=ARG/FAC
$BF7B (49019)  FAC=ARG^FAC
```

Routine $B475 (46197) haalt de lengte van de string in de accu. LSB en MSB staan in het X- en Y-register.

```
$0071 113     vector low (wijzer voor polynoominterpretatie)
$0072 114     vector high (Ze fungeren als wijzer op
              polynoomcoëfficiënt ($E043) en als
              wijzer op polynoomgraad ($E059))
$0073 115     CHRGET-routine (assembler ROM-listing)
$0074 114
$0075 117     $0073 INC $7A
$0076 118     $0075 BNE $0079 wijzer in BASIC-tekst verhogen
$0077 119     $0077 INC $7B
$0078 120     $0079 LDA $HHLL
$0079 121
$007A 122     programmawijzer low
$007B 123     programmawijzer high
$007C 124
$007D 125     $007C CMP "$3A ":"
$007E 126     $007E BSC $008A
$007F 127     $0080 CMP "$20 " " (spatie overslaan)
$0080 128     $0082 BEQ $0073
$0081 129     $0084 SEC
$0082 130     $0085 SBC "$30
$0083 131     $0087 SEC
$0084 132     $0088 SBC "$D0
```

$0085	133		$008A RTS
$0086	134		
$0087	135		De CHRGET-routine ligt eigenlijk in het ROM
$0088	136		vanaf $E3A2, maar wordt tijdens de RESET in
$0089	137		het RAM vanaf $0073 gekopieerd. De routine
$008A	138		functioneert alleen in het RAM-bereik. X- en Y-register worden niet aangetast carry=0: inhoud van gelezen byte ligt in $30-$39 (cijfers in ASCII-code) zero=1: teken=$00 of $3A (":")
$008B	139		laatste RND-waarde
$008C	140		idem
$008D	141		idem
$008E	142		idem
$008F	143		idem
$0090	144		status ST (bijvoorbeeld bij IEC-busroutines)
$0091	145	10	systeem-flag voor STOP-toets
$0092	146		tijdconstante voor cassette (ingesteld door $F92C)
$0093	147		LOAD/VERIFY-flag ($00=LOAD, $01=VERIFY) Ingesteld door LOAD-routine; LOAD-instructie stelt $0A (10) in
$0094	148		flag voor gereset teken bij IEC-OUT
$0095	149		gereset teken (buffer)
$0096	150		end-of-tape (EOT) gevonden (flag)
$0097	151		tussengeheugen voor registers
$0098	152		aantal open files POKE 152,0 sluit alle files POKE 153,12 verhindert openen van files
$0099	153	01	actief invoerapparaat (1=toetsenbord) POKE 153,2 verhindert invoer via toetsenbord
$009A	154	03	actief uitvoerapparaat (3=beeldscherm) POKE 154,1 betekent hetzelfde als bij adres 19 (INPUT)
$009B	155	00	pariteitsbyte van band
$009D	156		flag voor byte ontvangen
$009E	157		direkte modus? ($00=programma, $80=RUN) met GOTO een programma starten levert $80 op
$009E	158		bandfilter 1 proefsom
$009F	159		bandfilter 2 correctie
$00A0	160		TIME 3 (overdracht TIME 2)
$00A1	161		TIME 2 (overdracht TIME 1)
$00A2	162		TIME 1 (teller voor TI en TI$)
$00A3	163		bitteller (telt bij $FB97 de 8 bits op van een uit te voeren byte)
$00A4	164		teller voor cycli ($A3=8, dan $A4=0)
$00A5	165		aftel-teller op band zetten
$00A6	166		wijzer in cassettebuffer (wordt een teken in de cassettebuffer geschreven, dan wordt dit adres

			1 hoger en wijst op de volgende vrije plaats. Is de buffer vol (192 tekens), dan wordt de zero-flag gezet ($F80D).
$00A7	167		tussengeheugen voor LOAD/SAVE bij cassette
$00A8	168		idem
$00A9	169		idem
$00AA	170		idem
$00AB	171		idem
$00AC	172		wijzer voor bandbuffer en scrolling
$00AD	173		MSB voor bandbuffer en scrolling
$00AE	174		LSB van eindadres van geladen programma
$00AF	175		MSB van eindadres van geladen programma
$00B0	176	00	tijdconstanten voor band
$00B1	177		idem
$00B2	178		vector low (verschuiven van cassettebuffer)
$00B3	179		vector high
$00B4	180		bitteller voor band
$00B5	181		volgende bit voor RS-232
$00B6	182		buffer voor uit te voeren byte
$00B7	183		lengte van programmanaam
$00B8	184		laatst gebruikte filenummer
$00B9	185		laatst gebruikte secundaire adres
$00BA	186		laatst gebruikte devicenummer
$00BB	187		vector low (op opgeslagen programmanaam)
$00BC	188		vector high bij diskette
$00BD	189		tussengeheugen voor seriële in/uitvoer
$00BE	190		aantal nog te lezen/schrijven blocks (routine vanaf $FBCD)
$00BF	191		seriële woordbuffer
$00C0	192		flag voor cassettemotor
$00C1	193		vector low (beginadres van in/uitvoer)
$00C2	194		vector high (wijst op $A0000); voor SAVE-routine
$00C3	195		eindadres voor in/uitvoer van beeldscherm
$00C4	196		MSB voor in/uitvoer van beeldscherm
$00C5	197		nummer van ingedrukte toets (64=geen toets) 10 IF PEEK(197)=64 THEN 10 wacht op toetsdruk (zie ook tabel in bijlage)
$00C6	198		aantal toetsen, die uit de toetsenbordbuffer moeten worden uitgevoerd (zie toetsenbordbuffer)
$00C7	199		reverse flag (RVS) (1=reverse, 0=normaal) POKE 199,1: PRINT "DEMO"
$00C8	200		wijzer op regeleinde voor invoer
$00C9	201		cursorregel voor invoer (fungeert alleen als buffer)
$00CA	202		cursorkolom voor invoer (fungeert alleen als buffer)
$00CB	203		nummer van ingedrukte toets (64=geen toets)

		Schijnbaar dezelfde functie als adres 197, maar voor de computer een uitgestelde controle op tussentijds ingedrukte toetsen.
$00CC	204	cursor flag (0=cursor aan, 1=cursor uit) Staat toe, dat cursor in programma wordt ingeschakeld.
$00CD	205	teller voor knipperen van cursor
$00CE	206	teken in cursorpositie
$00CF	207	inschakel-flag (1=cursor zichtbaar, 0=cursor onzichtbaar) 10 IF PEEK(207) THEN 10 wacht totdat cursor zich in uit-stand bevindt
$00D0	208	invoer-flag (bijvoorbeeld $E65F, $F16A)
$00D1	209	vector low in actuele video-RAM
$00D2	210	vector high
$00D3	211	invoer van cursorkolom voor invoercursor zie voor cursorregel adres 214. Oproep van SET-routine: SYS 58732.
$00D4	212	aanhalingsflag (te gebruiken bij bijvoorbeeld de ESCAPE-routine. POKE 212,1: PRINT ... voert ook stuurtekens uit)
$00D5	213	lengte van beeldschermregel (39/79)
$00D6	214	invoer van cursorregel (zie adres 211)
$00D7	215	diverse doelen: laatste toets, buffer, proefsom
$00D8	216	aantal openstaande inserts
$00D9	217	MSB van begin van schermregels

tot

$00F0	240	idem
$00F1	241	onechte beeldschermregel
$00F2	242	kenmerk van beeldschermregels
$00F3	243	vector low (wijzer in actuele kleur-RAM)
$00F4	244	vector high (vanaf $D800)
$00F5	245	vector low (decodeertabel voor toetsenbord)
$00F6	246	vector high (wijzer: $ED81 (60289))
$00F7	247	vector low (RS-232 invoerbuffer)
$00F8	248	vector high
$00F9	249	vector low (RS-232 uitvoerbuffer)
$00FA	250	vector high
$F0FB	251	nog niet in gebruik
$F0FC	252	idem
$F0FD	253	idem
$F0FE	254	idem
$F0FF	255	begin van buffer voor reëel naar ASCII

EINDE ZEROPAGE

11.4 Belangrijke adressen van de volgende pages

In de toetsenbordbuffer ($0277-$0280/631-640) worden tekens (maximaal 10) opgeslagen, als bijvoorbeeld de ingedrukte toetsen niet kunnen worden verwerkt. De uitvoer van de tekens is mogelijk als het aantal in adres 198 staat. De uitvoer vindt plaats als het programma is afgelopen, dus al in de direkte modus. Op deze manier kunnen BASIC-regels in een lopend programma worden ingevoegd (zie paragraaf 2.11).

De kleur van de tekens kan worden veranderd door de inhoud van $0286 (646: geheugen voor tekenkleur) te wijzigen:

```
10 A=INT(RND(1)*15)
20 POKE 646,A : REM TOEVALLIGE KLEUR
30 PRINT"*";
40 GOTO 10
```

0 = zwart	8 = oranje
1 = wit	9 = bruin
2 = rood	10 = lichtrood
3 = cyaan	11 = grijs 1
4 = paars	12 = grijs 2
5 = groen	13 = lichtgroen
6 = blauw	14 = lichtblauw
7 = geel	15 = grijs 3

```
PRINT PEEK(648)*256
```

648 ($0288) is de MSB van het video-RAM. De PRINT-PEEK-instructie berekent het actuele beginadres van het beeldschermgeheugen. Meestal is dat 1024. Met bit 4 en 7 van adres 53272 en bit 0 en 1 van 56576 kan het beeldschermgeheugen worden verschoven (zie hoofdstuk 5). Adres 650 ($028A) regelt de repeatfunctie voor het toetsenbord.

```
POKE 650,   0  repeat alleen voor stuurtekens
POKE 650,  64  repeat uit
POKE 650, 128  repeat voor alle toetsen
```

De teller voor repeatvertraging ($028C/652) bepaalt de tijd tussen de uitvoer van het eerste teken en de volgende tekens. Bit 0, 1 en 2 van $028D-$028E (653-654) bepalen of de flag voor de SHIFT-, C=-, of CTRL-toets is gezet. $0291 (657: flag bij blokkade van SHIFT/C=) bepaalt of SHIFT/C= de omschakeling van de ene naar de andere karakterset toestaat.

11.5 Het opslaan van variabelen

Het volgende overzicht maakt duidelijk op welke plaats in het geheugen welke variabelen worden opgeslagen

```
|---------|   -begin van BASIC-geheugen ($2B, $2C) $0800
|         |
|         |
|    A    |   -bereik A: BASIC-programma
|         |
|         |
|---------|   -begin van variabelen ($2D, $2E)
|    B    |   -bereik B: opslag van variabelen
|---------|   -begin van arrays ($2F, $30)
|    C    |   -bereik C: opslag van arrayvariabelen
|---------|   -einde van arrays ($31, $32)
|         |
|         |
|         |
|         |   -vrij BASIC-geheugen
|/\/\/\/\/|

|\/\/\/\/\|
|         |   -vrij BASIC-geheugen
|         |
|         |
|         |
|---------|   -begin van strings (gaat naar beneden $33, $34)
|         |   -opgeslagen strings
|         |
|---------|   -einde van BASIC-geheugen ($37, $38)
|         |   -$9FFF
|---------|
```

In bereik A staat het BASIC-programma. Reële en integere variabelen worden opgeslagen in bereik B, evenals DEF FN. In C staan de arrays, de variabelenvelden. Vanaf het einde van het BASIC-geheugen staan de strings. Alle bereiken worden bepaald door wijzers die de interpreter gewoonlijk automatisch instelt. De wijzers kunt u ook wijzigen. Speciaal BASIC-start en -einde komen daarvoor in aanmerking.

43-44	BASIC-start
45-46	begin van variabelen
47-48	begin van array
49-50	einde van array
51-52	begin van string
53-54	hulpwijzer
55-56	BASIC-einde

Wordt vector 45-46 verhoogd, dan lijkt het programma langer als het wordt opgeslagen. De SAVE-routine slaat alles op tussen de grenzen die in 43-44 en 45-46 staan.

Omdat de arrays over een eigen geheugenplaats beschikken, zijn er in totaal nog 4 variabeletypes:

(1) reële variabelen (met namen als A, B, CD, enz.)
(2) stringvariabelen (gekenmerkt door $)
(3) integere variabelen (gekenmerkt door %)
(4) functies (FN)

Een variabelenaam bestaat uit twee bytes, ook als er maar een naamteken is gegeven. Omdat negatieve variabelenamen niet bestaan, blijven er twee zevende bits (4 mogelijkheden) over, die de vier variabeletypes onderscheiden. In bereik B is voor elke variabele 7 bytes gereserveerd.

BIT 7 VAN:

NAAMTEKEN 1	NAAMTEKEN 2	VARIABELETYPE
0	0	reëel
0	1	string
1	0	functie
1	1	integer

De 7 bytes:

BYTE	1	2	3	4	5	6	7	VARIABELE
	NT1	NT2	opgeslagen waarde binair					reëel
	NT1	NT2	lengte	LO	HI	0	0	string
	NT1	NT2	HI	LO	0	0	0	integer

NT1 = eerste naamteken
NT2 = tweede naamteken
LO en HI = lo- en hi-adres van variabele-inhoud

FN is een uitzondering. Deze functie beslaat 14 bytes:

BYTE	1	2	3	4	5	6	7	8	9
	NT1F	NT2F	LO	HI	LOA	HIA	ASCD	NT1A	NT2A

BYTE	10	11	12	13	14
	waarde van argumentvariabele				

NT1F/NT2F = eerste en tweede naamteken van functie
LO/HI = lo- en hi-adres van functie
LOA/HIA = lo- en hi-adres van argument
NT1A/NT2A = eerste en tweede naamteken van argument

11.6 Lijst van belangrijke wijzers

Veel ROM-routines worden via wijzers in het RAM aangesproken. Door die wijzers te veranderen kunnen andere routines worden gebruikt zonder dat het ROM in het RAM hoeft te worden gekopieerd.

LSB	MSB	SPRONGADRES	ROUTINE
3	4		verandering van reëel in geheel
5	6		verandering van geheel in reëel
23	24		wijzer op laatst gebruikte string
34	35		wijzer voor eigen gebruik
36	37		idem
43	44		BASIC-start (byte 1=0)
45	46		begin van variabelen
47	48		begin van arrays
49	50		einde van arrays
51	52		begin van strings (gaan naar beneden)
53	54		hulpwijzers voor strings
55	56		einde van BASIC
61	62		volgende instructie voor CONT
65	66		volgende DATA-element
85	86		sprongvector voor functies
122	123		programmawijzer voor CHRGET-routine
178	179		beginadres van cassettebuffer

187	188		wijzer op programmanaam
209	210		wijzer op actuele BS-regel
243	244		wijzer in actuele kleur-RAM
245	246		decodeertabel voor toetsenbord
247	248		RS-232 invoerbuffer
249	250		RS-232 uitvoerbuffer
655	656		decodering van toetsenbord
768	769	$E38B	BASIC warme start
770	771	$A438	invoer van regels
772	773	$A57C	veranderen in interpretercode
774	775	$A71A	LIST (begrijpelijke tekst)
776	777	$A7E4	adres van BASIC-instructie halen
778	779	$AE86	term interpreteren
785	786	$B248	USR-wijzer
788	789	$EA31	interrupt(IRQ)-vector
790	791	$FE66	BREAK-vector
792	793	$FE47	niet maskeerbare interrupt (NMI)
794	795	$F34A	OPEN
796	797	$F291	CLOSE
798	799	$F20E	CHKIN
800	801	$F250	CKOUT
802	803	$F333	CLRCH
804	805	$F157	INPUT
806	807	$F1CA	OUTPUT
808	809	$F6ED	STOP
810	811	$F13E	GET
812	813	$F32F	CLALL
814	815	$FE66	warme start
816	817	$F4A5	LOAD
818	819	$F5ED	SAVE

Wijzers inlezen doet u met

```
PRINT PEEK(low)+256*PEEK(high)
```

Sommige wijzers kunnen alleen gewijzigd worden, als de interrupt is verhinderd:

```
POKE 56334,0  interrupt uit
POKE 56334,1  interrupt aan
```

12 BIJLAGE

12.1 De tabellen

In het handboek voor de Commodore 64 zijn belangrijke tabellen vergeten. In het volgende overzicht is een aantal kleine tabellen samengevoegd tot een grote. Op die manier kunt u betekenissen en waarden die bij de getallen horen, sneller vinden. Alleen het inlezen van de joysticks is een aparte tabel. In tabel 1 staan de drie meest gebruikte getallenstelsels (decimaal, hexadecimaal en binair), machinetaalinstructies (MNEMONICS), BASIC-instructies (TOKENS) en de decodeertabel voor het toetsenbord. In BASIC worden bijna altijd decimale getallen gebruikt, in machinetaal alleen maar hexadecimale. Om te begrijpen hoe de computer werkt, moet u met het binaire stelsel vertrouwd zijn. In de kolom voor MNEMONICs staat het formaat van machinetaalinstructies. Elke z is een hexadecimaal getal. In het geheugen worden BASIC-instructies (kolom 5) in getallen omgezet. De laatste kolom bevat de decodeertabel voor het toetsenbord. Deze waarden worden bijvoorbeeld in adres 197 of 203 uitgevoerd.

12.2 Omrekeningstabel

DEC.	HEX.	BIN.	MNEMONIC	TOKEN	TOETSOPDRUK
0	$00	%00000000	BRK		Inst Del
1	$01	%00000001	ORA (zz,X)		RETURN
2	$02	%00000010			Cursor right
3	$03	%00000011			F7
4	$04	%00000100			F1
5	$05	%00000101	ORA zz		F3
6	$06	%00000110	ASL zz		F5
7	$07	%00000111			Cursor down
8	$08	%00001000	PHP		3
9	$09	%00001001	ORA #zz		W
10	$0A	%00001010	ASL		A
11	$0B	%00001011			4
12	$0C	%00001100			Z
13	$0D	%00001101	ORA zzzz		S
14	$0E	%00001110	ASL zzzz		E
15	$0F	%00001111			SHIFT links
16	$10	%00010000	BPL zzzz		5
17	$11	%00010001	ORA (zz),y		R
18	$12	%00010010			D
19	$13	%00010011			6
20	$14	%00010100			C
21	$15	%00010101	ORA zz,X		F
22	$16	%00010110	ASL zz,X		T
23	$17	%00010111			X
24	$18	%00011000	CLC		7
25	$19	%00011001	ORA zzzz,y		Y
26	$1A	%00011010			G
27	$1B	%00011011			8
28	$1C	%00011100			B
29	$1D	%00011101	ORA zzzz,X		H

DEC.	HEX.	BIN.	MNEMONIC	TOKEN	TOETSOPDRUK
30	$1E	%00011110	ASL zzzz,X		U
31	$1F	%00011111			V
32	$20	%00100000	JSR zzzz		9
33	$21	%00100001	AND (zz,X)		I
34	$22	%00100010			J
35	$23	%00100011			O
36	$24	%00100100	BIT zz		M
37	$25	%00100101	AND zz		K
38	$26	%00100110	ROL zz		0
39	$27	%00100111			N
40	$28	%00101000	PLP		+
41	$29	%00101001	AND #zz		P
42	$2A	%00101010	ROL		L
43	$2B	%00101011			-
44	$2C	%00101100	BIT zzzz		.
45	$2D	%00101101	AND zzzz		:
46	$2E	%00101110	ROL zzzz		
47	$2F	%00101111			'
48	$30	%00110000	BMI zzzz		pond
49	$31	%00110001	AND (zz),y		*
50	$32	%00110010			;
51	$33	%00110011			CLR HOME
52	$34	%00110100			SHIFT rechts
53	$35	%00110101	AND zz,X		=
54	$36	%00110110	ROL zz,X		?
55	$37	%00110111			-
56	$38	%00111000	SEC		1
57	$39	%00111001	AND zzzz,y		pijl
58	$3A	%00111010			CTRL
59	$3B	%00111011			2

DEC.	HEX.	BIN.	MNEMONIC	TOKEN	TOETSOPDRUK
60	$3C	%00111100			SPACE
61	$3D	%00111101	AND zzzz,X		commodore
62	$3E	%00111110	ROL zzzz,X		Q
63	$3F	%00111111			RUN STOP
64	$40	%01000000	RTI		
65	$41	%01000001	EOR (zz,X)		
66	$42	%01000010			
67	$43	%01000011			
68	$44	%01000100			
69	$45	%01000101	EOR zz		
70	$46	%01000110	LSR zz		
71	$47	%01000111			
72	$48	%01001000	PHA		
73	$49	%01001001	EOR #zz		
74	$4A	%01001010	LSR		
75	$4B	%01001011			
76	$4C	%01001100	JMP zzzz		
77	$4D	%01001101	EOR zzzz		
78	$4E	%01001110	LSR zzzz		
79	$4F	%01001111			
80	$50	%01010000	BVC zzzz		
81	$51	%01010001	EOR (zz),y		
82	$52	%01010010			
83	$53	%01010011			
84	$54	%01010100			
85	$55	%01010101	EOR zz,X		
86	$56	%01010110	LSR zz,X		
87	$57	%01010111			
88	$58	%01011000	CLI		
89	$59	%01011001	EOR zzzz,y		

DEC.	HEX.	BIN.	MNEMONIC	TOKEN	TOETSOPDRUK
90	$5A	%01011010			
91	$5B	%01011011			
92	$5C	%01011100			
93	$5D	%01011101	EOR zzzz,X		
94	$5E	%01011110	LSR zzzz,X		
95	$5F	%01011111			
96	$60	%01100000	RTS		
97	$61	%01100001	ADC (zz,X)		
98	$62	%01100010			
99	$63	%01100011			
100	$64	%01100100			
101	$65	%01100101	ADC zz		
102	$66	%01100110	ROR zz		
103	$67	%01100111			
104	$68	%01101000	PLA		
105	$69	%01101001	ADC #zz		
106	$6A	%01101010	ROR		
107	$6B	%01101011			
108	$6C	%01101100	JMP (zzzz)		
109	$6D	%01101101	ADC zzzz		
110	$6E	%01101110	ROR zzzz		
111	$6F	%01101111			
112	$70	%01110000	BVS zzzz		
113	$71	%01110001	ADC (zz),y		
114	$72	%01110010			
115	$73	%01110011			
116	$74	%01110100			
117	$75	%01110101	ADC zz,X		
118	$76	%01110110	ROR zz,X		
119	$77	%01110111			

DEC.	HEX.	BIN.	MNEMONIC	TOKEN	TOETSOPDRUK
120	$78	%01111000	SEI		
121	$79	%01111001	ADC zzzz,y		
122	$7A	%01111010			
123	$7B	%01111011			
124	$7C	%01111100			
125	$7D	%01111101	ADC zzzz,X		
126	$7E	%01111110	ROR zzzz,X		
127	$7F	%01111111			
128	$80	%10000000		END	
129	$81	%10000001	STA (zz,X)	FOR	
130	$82	%10000010		NEXT	
131	$83	%10000011		DATA	
132	$84	%10000100	STY zz	INPUT#	
133	$85	%10000101	STA zz	INPUT	
134	$86	%10000110	STX zz	DIM	
135	$87	%10000111		READ	
136	$88	%10001000	DEY	LET	
137	$89	%10001001		GOTO	
138	$8A	%10001010	TAX	RUN	
139	$8B	%10001011		IF	
140	$8C	%10001100	STY zzzz	RESTORE	
141	$8D	%10001101	STA zzzz	GOSUB	
142	$8E	%10001110	STX zzzz	RETURN	
143	$8F	%10001111		REM	
144	$90	%10010000	BCC zzzz	STOP	
145	$91	%10010001	STA (zz),y	ON	
146	$92	%10010010		WAIT	
147	$93	%10010011		LOAD	
148	$94	%10010100	STY zz,X	SAVE	
149	$95	%10010101	STA zz,X	VERIFY	

DEC.	HEX.	BIN.	MNEMONIC	TOKEN	TOETSOPDRUK
150	$96	%10010110	STY zz,y	DEF	
151	$97	%10010111		POKE	
152	$98	%10011000	TYA	PRINT#	
153	$99	%10011001	STA zzzz,y	PRINT	
154	$9A	%10011010	TXS	CONT	
155	$9B	%10011011		LIST	
156	$9C	%10011100		CLR	
157	$9D	%10011101	STA zzzz,X	CMD	
158	$9E	%10011110		SYS	
159	$9F	%10011111		OPEN	
160	$A0	%10100000	LDY #zz	CLOSE	
161	$A1	%10100001	LDA (zz,X)	GET	
162	$A2	%10100010	LDX #zz	NEW	
163	$A3	%10100011		TAB(	
164	$A4	%10100100	LDY zz	TO	
165	$A5	%10100101	LDA zz	FN	
166	$A6	%10100110	LDX zz	SPC(	
167	$A7	%10100111		THEN	
168	$A8	%10101000	TAY	NOT	
169	$A9	%10101001	LDA #zz	STEP	
170	$AA	%10101010	TAX	+	
171	$AB	%10101011		-	
172	$AC	%10101100	LDY zzzz	*	
173	$AD	%10101101	LDA zzzz	/	
174	$AE	%10101110	LDX zzzz	^	
175	$AF	%10101111		AND	
176	$B0	%10110000	BCS zzzz	OR	
177	$B1	%10110001	LDA (zz),y	:	
178	$B2	%10110010		=	
179	$B3	%10110011		;	

DEC.	HEX.	BIN.	MNEMONIC	TOKEN	TOETSOPDRUK
180	$B4	%10110100	LDY zz,X	SGN	
181	$B5	%10110101	LDA zz,X	INT	
182	$B6	%10110110	LDX zz,y	ABS	
183	$B7	%10110111		USR	
184	$B8	%10111000	CLV	FRE	
185	$B9	%10111001	LDA zzzz,y	POS	
186	$BA	%10111010	TSX	SQR	
187	$BB	%10111011		RND	
188	$BC	%10111100	LDY zzzz,X	LOG	
189	$BD	%10111101	LDA zzzz,X	EXP	
190	$BE	%10111110	LDX zzzz,y	COS	
191	$BF	%10111111		SIN	
192	$C0	%11000000	CPY #zz	TAN	
193	$C1	%11000001	CMP (zz,X)	ATN	
194	$C2	%11000010		PEEK	
195	$C3	%11000011		LEN	
196	$C4	%11000100	CPY zz	STR$	
197	$C5	%11000101	CMP zz	VAL	
198	$C6	%11000110	DEC zz	ASC	
199	$C7	%11000111		CHR$	
200	$C8	%11001000	INY	LEFT$	
201	$C9	%11001001	CMP #zz	RIGHT$	
202	$CA	%11001010	DEX	MID$	
203	$CB	%11001011		GO	
204	$CC	%11001100	CPY zzzz		
205	$CD	%11001101	CMP zzzz		
206	$CE	%11001110	DEC zzzz		
207	$CF	%11001111			
208	$D0	%11010000	BNE zzzz		
209	$D1	%11010001	CMP (zz),y		

DEC.	HEX.	BIN.	MNEMONIC	TOKEN	TOETSOPDRUK
210	$D2	%11010010			
211	$D3	%11010011			
212	$D4	%11010100			
213	$D5	%11010101	CMP zz,X		
214	$D6	%11010110	DEC zz,X		
215	$D7	%11010111			
216	$D8	%11011000	CLC		
217	$D9	%11011001	CMP zzzz,y		
218	$DA	%11011010			
219	$DB	%11011011			
220	$DC	%11011100			
221	$DD	%11011101	CMP zzzz,X		
222	$DE	%11011110	DEC zzzz,X		
223	$DF	%11011111			
224	$E0	%11100000	CPX #zz		
225	$E1	%11100001	SBC (zz,X)		
226	$E2	%11100010			
227	$E3	%11100011			
228	$E4	%11100100	CPX zz		
229	$E5	%11100101	SBC zz		
230	$E6	%11100110	INC zz		
231	$E7	%11100111			
232	$E8	%11101000	INX		
233	$E9	%11101001	SBC #zz		
234	$EA	%11101010	NOP		
235	$EB	%11101011			
236	$EC	%11101100	CPX zzzz		
237	$ED	%11101101	SBC zzzz		
238	$EE	%11101110	DEC zzzz		
239	$EF	%11101111			

DEC.	HEX.	BIN.	MNEMONIC	TOKEN	TOETSOPDRUK
240	$F0	%11110000	BEQ zzzz		
241	$F1	%11110001	SBC (zz),y		
242	$F2	%11110010			
243	$F3	%11110011			
244	$F4	%11110100			
245	$F5	%11110101	SBC zz,X		
246	$F6	%11110110	INC zz,X		
247	$F7	%11110111			
248	$F8	%11111000	SED		
249	$F9	%11111001	SBC zzzz,y		
250	$FA	%11111010			
251	$FB	%11111011			
252	$FC	%11111100			
253	$FD	%11111101	SBC zzzz,X		
254	$FE	%11111110	INC zzzz,X		
255	$FF	%11111111		PI	

12.3 Tabel van device-nummers

NUMMER	APPARAAT
0	toetsenbord
1	tape (cassette)
2	RS-232 interface
3	beeldscherm
4	printer
5	(optioneel) printer
6	printer/plotter
7	vrij
8	diskdrive
9	(optioneel) diskdrive
10	idem
11	idem
12	idem
13	idem
14	idem
15	idem

Ook deze tweede band wekt de experimenteerlust op.
In het boek zult u een omvangrijk hoofdstuk over het beveiligen van programma's aantreffen, pointers en hun gebruik, meer informatie over de interrupts, het kopieëren van het besturingssysteem naar RAM en veel nuttige toepassingen. Wie graag zijn of haar Commodore-64 programmeert, zal dit boek goed kunnen gebruiken.

ISBN 90 229 3337 7

Commodore Bibliotheek 12

DATA BECKER
NEDERLANDS*

Tips en Trucs voor de Commodore 64 deel 2

CB12

DATA BECKER
NEDERLANDS*